

***Übungen zu  
Rechnerarchitektur  
– Tomasulo's Algorithm –***

Sommersemester 2017

Jian-Jia Chen

Folien von Michael Engel, Björn Bönninghoff,  
und Olaf Neugebauer

Technische Universität Dortmund

Lehrstuhl Informatik 12

Entwurfsautomatisierung für Eingebettete Systeme

# Verfahren von Tomasulo: Prinzip

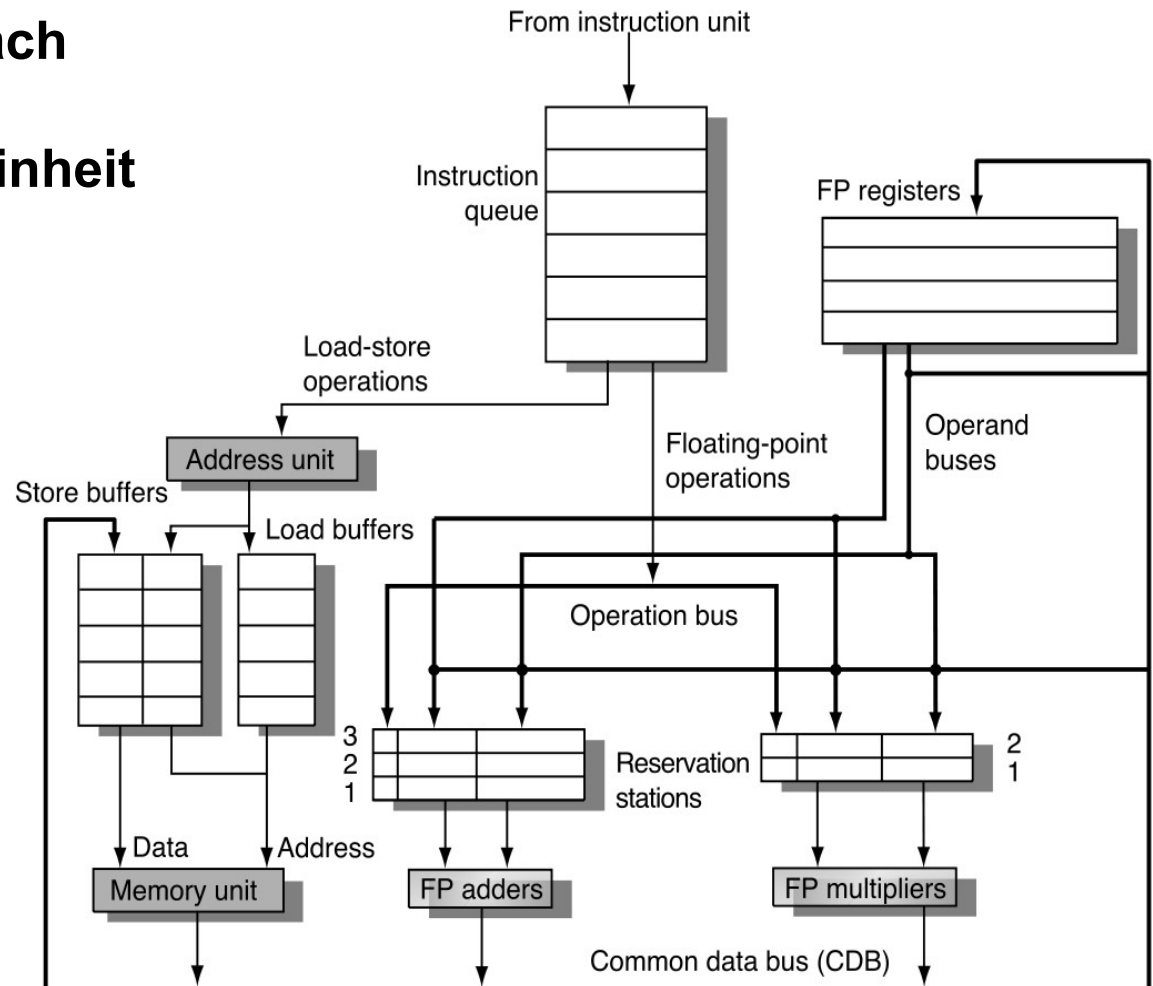
- **Wichtige Hardwarestruktur: *Reservation Stations***
  - Zugeordnet zu funktionalen Einheiten (i.d.R. eine pro Einheit) □
- **Arbeitsweise von *Reservation Stations*:**
  - Puffern Operanden für Befehle (sobald verfügbar geladen)  
→ Müssen nicht aus Registern gelesen werden!
  - Ausstehende Operanden verweisen auf *Reservation Station*, die Eingabe bereitstellen wird
  - Bei aufeinander folgenden Schreibzugriffen auf Register:  
Nur letzter für Aktualisierung des Inhalts verwendet
- **Falls  $|Reservation Stations| > |Register|$  ☞ Namenskonflikte lösbar, die Compiler nicht behandeln kann!**

## Verfahren von Tomasulo: Prinzip (2)

- Verwendung von *Reservation Stations* anstelle des zentralen Registersatzes bewirkt 2 wichtige Eigenschaften:
  - Konfliktdetektion und Ausführungskontrolle verteilt
    - Informationen in *Reservation Stations* bei den funktionellen Einheiten bestimmen, wann Ausführung eines Befehls möglich
  - Ergebnisse werden direkt zu den funktionellen Einheiten (in jeweiliger *Reservation Station*) weitergereicht
    - Erweiterte Form des *Forwarding*
    - Realisiert implizit *Register Renaming*
    - Möglich durch gemeinsamen Ergebnisbus (*common data bus*)  
→ Muss ggf. mehrfach vorhanden sein, falls mehr als ein Ergebnis pro Takt weiterzuleiten ist

# Verfahren von Tomasulo: Prinzip (3)

- Grundstruktur einer nach dem Tomasulo-Prinzip realisierten MIPS-FP-Einheit
  
- Nur realisiert:
  - FP-Ops. und
  - Load/Store
  
- **Load/Store Buffer** vergleichbar **Reservation Stations**



# Bearbeitungsphasen (1)

## ■ Ausführung eines Befehls in 3 Bearbeitungsphasen:

### 1. *Issue*

- Befehl aus Befehlswarteschlange lesen (IF bereits erfolgt)
- In *Reservation Station* der benötigten funktionellen Einheit eintragen, sofern eine frei
- Operandenwerte aus Registern lesen (sofern dort vorhanden)
- Falls Operanden erst erzeugt, Referenz auf erzeugende *Reservation Station* eintragen (und Entstehen „beobachten“)
- *Reservation Stations* wie weitere Register behandelt  
→ WAR und WAW-Konflikte auflösbar  
(*Register Renaming* damit implizit erreicht)

# Bearbeitungsphasen (2)

## 1. Execution

- *Reservation Stations* beobachten Ergebnis-Bus, sofern Operanden noch nicht verfügbar
- Neu berechnete Operanden in *Reservation Stations* eintragen
- Sobald alle Operanden verfügbar: Abarbeitung des Befehls in korrespondierender fkt. Einheit starten
- Hinweise:
  - RAW-Konflikte durch datengetriebenes Starten der Befehlsausführung gelöst („Warten“ = *Pipeline-Stalls*)
  - > 1 Befehle können gleichzeitig ausführbereit werden

# Bearbeitungsphasen (2)

## 1. *Write Results*

- Ergebnisse werden via Ergebnis-Bus in wartende *Reservation Stations* und ggf. in Register geschrieben
- Schreiben in Register: Nur wenn noch kein logisch nachfolgender Schreibzugriff auf das selbe Register vorgemerkt (abschließendes WAW-Handling)

# Verfahren von Tomasulo: Datenstrukturen

- **Notation in Anlehnung an *Scoreboarding***
  - 1. Befehlsstatus: Phase (*Issue, ... Write Results*) in der sich Befehl befindet
  - 2. Status der *Reservation Stations*:
    - *Busy*: Einheit arbeitend oder nicht
    - *Op*: Aktuelle Operation
    - *Qj, Qk*: *Reservation Stations*, die ggf. Quelloperanden erzeugen (leer, wenn bereits in *Vj/Vk* verfügbar) □
    - *Vj, Vk*: Werte der Quelloperanden
  - Für *Load* bzw. *Store*-Puffer:
    - *A*: Effektive Adresse für Speichertransfer
  - 3. Register-Ergebnis-Status
    - *Qi*: *Reservation Station*, die Operation steuert, deren Ergebnis in aktuellem Register gespeichert werden soll



# Tomasulo-Algorithmus: Takt +0

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	
MUL.D	F0, F2, F4	+		
SUB.D	F8, F2, F6	+		
DIV.D	F10, F0, F6	+		
ADD.D	F6, F8, F2	+		
SUB.D	F8, F4, F0	+		

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	ja	Load					96+R3
Add1	ja	SUB		M[32+R2]	Load2		
Add2	ja	ADD			Add1	Load2	
Add3	ja	SUB	F4			Mult1	
Mult1	ja	MUL		F4	Load2		
Mult2	ja	DIV		M[32+R2]	Mult1		

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi	Mult1	Load2		Add2	Add3	Mult2	

# Tomasulo-Algorithmus: Takt +1

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+		
SUB.D	F8, F2, F6	+		
DIV.D	F10, F0, F6	+		
ADD.D	F6, F8, F2	+		
SUB.D	F8, F4, F0	+		

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	ja	SUB	M[96+R3]	M[32+R2]	-		
Add2	ja	ADD		M[96+R3]	Add1	-	
Add3	ja	SUB	F4				Mult1
Mult1	ja	MUL	M[96+R3]	F4	-		
Mult2	ja	DIV		M[32+R2]	Mult1		

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi	Mult1	-		Add2	Add3	Mult2	

# Tomasulo-Algorithmus: Takt +2

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	1	
SUB.D	F8, F2, F6	+	+	
DIV.D	F10, F0, F6	+		
ADD.D	F6, F8, F2	+		
SUB.D	F8, F4, F0	+		

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	ja	SUB					
Add2	ja	ADD		M[96+R3]	Add1		
Add3	ja	SUB	F4			Mult1	
Mult1	ja	MUL	M[96+R3]	F4			
Mult2	ja	DIV		M[32+R2]	Mult1		

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi	Mult1			Add2	Add3	Mult2	

# Tomasulo-Algorithmus: Takt +3

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	2	
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+		
ADD.D	F6, F8, F2	+		
SUB.D	F8, F4, F0	+		

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	ja	ADD	...	M[96+R3]	-		
Add3	ja	SUB	F4			Mult1	
Mult1	ja	MUL	M[96+R3]	F4			
Mult2	ja	DIV		M[32+R2]		Mult1	

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi	Mult1			Add2	Add3	Mult2	

# Tomasulo-Algorithmus: Takt +4

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+		
ADD.D	F6, F8, F2	+	+	
SUB.D	F8, F4, F0	+		

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	ja	ADD					
Add3	ja	SUB	F4				Mult1
Mult1	ja	MUL					
Mult2	ja	DIV		M[32+R2]		Mult1	

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi	Mult1			Add2	Add3	Mult2	

# Tomasulo-Algorithmus: Takt +5

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+		
ADD.D	F6, F8, F2	+	+	
SUB.D	F8, F4, F0	+		

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	ja	ADD					
Add3	ja	SUB	F4	...		-	
Mult1	nein						
Mult2	ja	DIV	...	M[32+R2]	-		

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi	-			Add2	Add3	Mult2	

# Tomasulo-Algorithmus: Takt +6

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	1	
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	ja	SUB					
Mult1	nein						
Mult2	ja	DIV	...	M[32+R2]			

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
Qi					Add3	Mult2	

# Tomasulo-Algorithmus: Takt +7

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	2	
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	+

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	ja	DIV	...	M[32+R2]			

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i> ...
Qi					-	Mult2



# Tomasulo-Algorithmus: Takt +8

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	3	
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	+

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	ja	DIV	...	M[32+R2]			

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i> ...
<i>Qi</i>						Mult2

# Tomasulo-Algorithmus: Takt +9

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	4	
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	+

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	ja	DIV	...	M[32+R2]			

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i> ...
Qi						Mult2

# Tomasulo-Algorithmus: Takt +10

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	5	
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	+

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	ja	DIV	...	M[32+R2]			

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i> ...
Qi						Mult2

# Tomasulo-Algorithmus: Takt +11

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	+	
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	+

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	ja	DIV					

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i> ...
Qi						Mult2

# Tomasulo-Algorithmus: Takt +12

## ■ Befehls-Status

<i>Befehl</i>		<i>Issue</i>	<i>Execute</i>	<i>Write Result</i>
L.D	F6, 32 (R2)	+	+	+
L.D	F2, 96 (R3)	+	+	+
MUL.D	F0, F2, F4	+	+	+
SUB.D	F8, F2, F6	+	+	+
DIV.D	F10, F0, F6	+	+	+
ADD.D	F6, F8, F2	+	+	+
SUB.D	F8, F4, F0	+	+	+

## ■ Status der Reservation Stations

<i>Name</i>	<i>Busy</i>	<i>Op</i>	<i>Vj</i>	<i>Vk</i>	<i>Qj</i>	<i>Qk</i>	<i>A</i>
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	nein						

## ■ Register-Ergebnis-Status

	<i>F0</i>	<i>F2</i>	<i>F4</i>	<i>F6</i>	<i>F8</i>	<i>F10</i>	<i>...</i>
<i>Qi</i>						-	

# Unterschiede Scoreboarding / Tomasulo

## Scoreboarding

Zentrale Instanz

4 Stufen: issue, read operands, execution, write result

Arbeitet ausschließlich auf phys. Registern

In jedem Takt wird zentral erfasst, welche Operanden bekannt sind

Antidaten- (WAR) und Ausgabeabhängigkeiten (WAW) immer durch Stalls gelöst

## Tomasulo

Konfliktdetektion und Ausführungskontrolle verteilt

3 Stufen: issue, execution, write result

Arbeitet auf logischen Registern (register renaming)

Ausstehende Operanden verweisen auf dezentral verteilte reservation stations

WAR und WAW gelöst

# Tomasulo: Multiple Issue

## ■ Multiple Issue

- Verbreiterung der Issue-Stufe
  - Sequentiell in „Mikrotakten“ oder
  - Parallel mit Betrachtung aller Abhängigkeiten gleichzeitig
- Verbreiterung des Common Data Bus zum Rückschreiben

# Tomasulo: Spekulation

## ■ Spekulation

- Einsatz nur sinnvoll mit Branch-Prediction (kennen wahrscheinliches Sprungziel vor echter Sprungauswertung)
- Spekulative Ausführung, Abschluß *in-order* durch neue *Commit*-Pipeline-Stufe
- *Reorder Buffer* speichert Register-Ergebnisse
  - Fehlerhaft vorherges. Branch: Ergebnisse verwerfen
  - Korrekt vorherges. Branch: Ergebnisse übernehmen
  - Spekulative Load/Stores: Verzögern



# Struktur

## Hardwarestruktur mit Reorder-Buffer (ROB):

- *Register Renaming* jetzt durch ROB realisiert
- *Reservation Stations* puffern Operation + Operanden zwischen *Issue* und Beginn der Ausführung

