Übungen zu Rechnerarchitektur – Scoreboarding and Tomasulo's Algorithm –

Sommersemester 2020

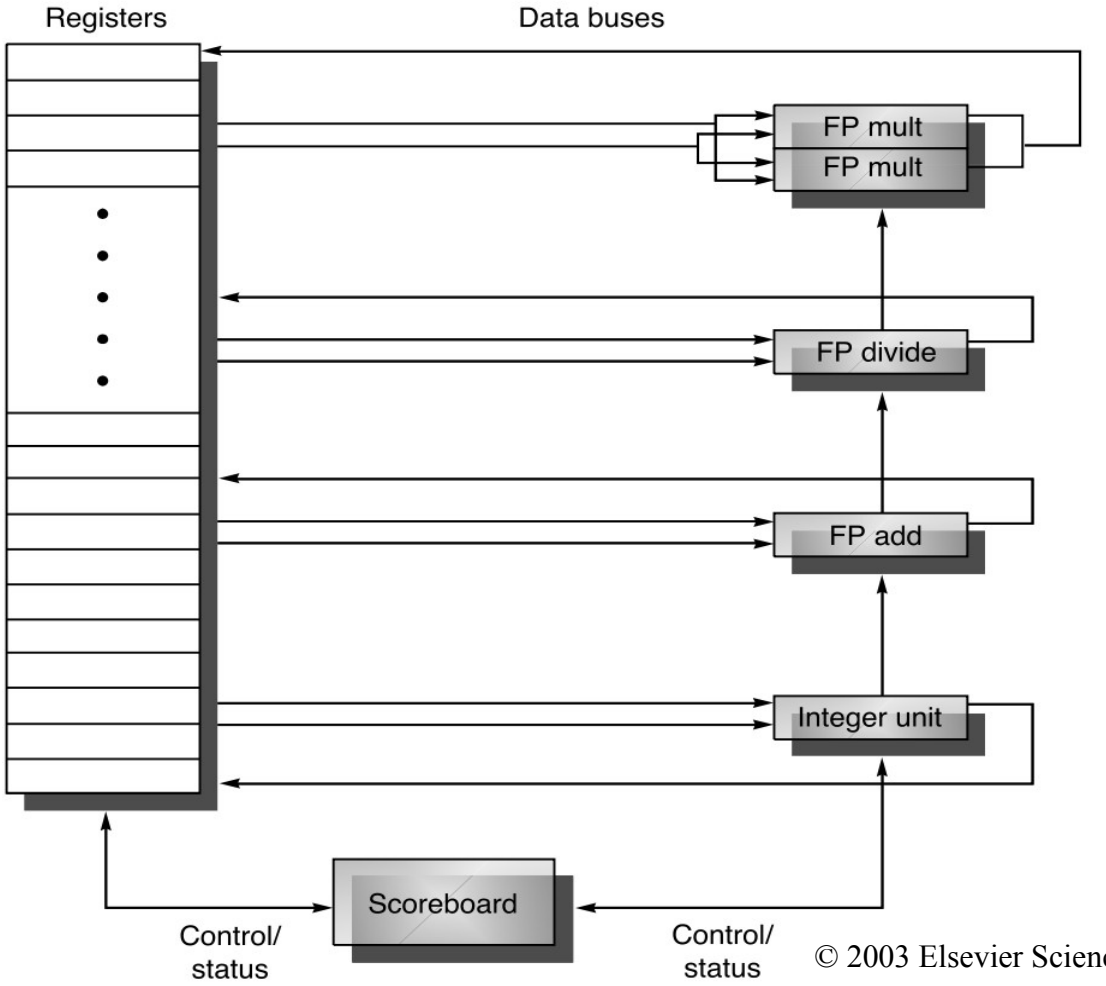
Jian-Jia Chen
Folien von Michael Engel, Björn Bönninghoff, und Olaf Neugebauer
Technische Universität Dortmund

Lehrstuhl Informatik 12

Entwurfsautomatisierung für Eingebettete Systeme

Scoreboarding

- **Struktur einer MIPS-*Pipeline* mit *Scoreboard***
- ***Scoreboard* kontrolliert funktionelle Einheiten**
- **Bussysteme erforderlich**



Funktion des *Scoreboards*

- **Scoreboard kontrolliert Befehlsausführung ab Befehlsholphase (IF) in 4 Phasen:**
- **1. *Issue***
  - Befehl wird ausgegeben falls
    - benötigte funktionale Einheit ist frei und
    - kein anderer aktiver Befehl (d.h. im *Scoreboard*) schreibt gleiches Zielregister ☞ verhindert WAW-Konflikte
  - Auswertung beginnt nicht, Befehl nur an funktionelle Einheit!
    - Ersetzt Teil der ID-Phase in klassischer Pipeline
    - Anhalten der *Pipeline* (*stall*) erfolgt:
      - Bei strukturellen Konflikten oder
      - WAW-Konflikt
      - ☞ kein weiterer Befehl wird ausgegeben!

Funktion des *Scoreboards*

- **2. *Read Operands* (RO)**
  - *Scoreboard* beobachtet Verfügbarkeit von Quelloperanden
    ☞sind verfügbar, wenn kein früher ausgegebener Befehl, der noch aktiv ist, diese schreiben wird (Datenflussprinzip)
  - Sobald Operanden verfügbar: *Scoreboard* signalisiert betreffender fkt. Einheit Beginn der Ausführungsphase
  - RAW-Konflikte (d.h. echte Datenabhängigkeiten) werden hier dynamisch aufgelöst
  - Befehle werden ggf. *out-of-order* zur Ausführung gebracht (*Issue* erfolgt nocht strikt *in-order*!)
  - *Scoreboarding* verwendet kein *Forwarding* (Komplexität!)
    ☞ Operanden werden bei Verfügbarkeit aus Register gelesen

Funktion des *Scoreboards*

- **3. *Execution***
  - Funktionelle Einheit beginnt Befehlsausführung nach Erhalt der Operanden
  - *Scoreboard* wird informiert, sobald Ergebnis vorliegt (noch in funktioneller Einheit, d.h. noch nicht im Register!)
  - Entspricht „klassischer" EX-Phase, ggf. >1 Zyklen lang
- **4. *Write Results* (WR)**
  - Ausführungseinheit meldet abgeschlossene Berechnung an *Scoreboard*
  - *Scoreboard* erteilt Berechtigung zum Abspeichern, wenn Zielregister nicht von aktiver Operation (in RO) noch benötigt ☞ Lösung von WAR-Konflikten
- **Abspeichern macht ggf. Befehl ausführungsbereit**
- **WR und RO dürfen nicht überlappen!**

Datenstrukturen

1. **Befehlsstatus**: Phase (*Issue, ... Write Results*), in der sich Befehl befindet
2. **Status der funktionelle Einheiten**, unterteilt in:
   - *Busy* : Einheit arbeitend oder nicht
   - Op : Aktuelle Operation
   - Fi : Zielregister
   - Fj, Fk : Quellregister
   - Qj, Qk : fkt. Einheiten, die ggf. Quellregisterinhalt erzeugen
   - Rj, Rk : *Flag*, ob Quelloperanden verfügbar aber, noch nicht in *Read Operands* gelesen
5. **Register-Ergebnis-Status**
   Welche funktionelle Einheit verfügbare Register schreibt (ggf. leer)

Scoreboarding: Takt +0

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	
MUL.D	**F0,F2,F4**	**+**			
SUB.D	**F8,F6,F2**	**+**			
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	ja	Load	F2	R3				nein	
Mult1	ja	Mult	F0	F2	F4	Int.		nein	ja
Mult2	nein								
Add	ja	Sub	F8	F6	F2		Int.	ja	nein
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1	Int			Add	Div

Scoreboarding: Takt +1

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**			
SUB.D	**F8,F6,F2**	**+**			
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein	-	-	-				-	
Mult1	ja	Mult	F0	F2	F4	-		ja	ja
Mult2	nein								
Add	ja	Sub	F8	F6	F2		-	ja	ja
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1	-			Add	Div

Scoreboarding: Takt +2

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**		
SUB.D	**F8,F6,F2**	**+**	**+**		
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	ja	Mult	F0	F2	F4			nein	nein
Mult2	nein								
Add	ja	Sub	F8	F6	F2			nein	nein
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1				Add	Div

Scoreboarding: Takt +3

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**1**	
SUB.D	**F8,F6,F2**	**+**	**+**	**1**	
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	ja	Mult	F0	F2	F4			nein	nein
Mult2	nein								
Add	ja	Sub	F8	F6	F2			nein	nein
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1				Add	Div

Scoreboarding: Takt +4

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**2**	
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	ja	Mult	F0	F2	F4			nein	nein
Mult2	nein								
Add	ja	Sub	F8	F6	F2			nein	nein
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1				Add	Div

Scoreboarding: Takt +5

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**3**	
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**				

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	ja	Mult	F0	F2	F4			nein	nein
Mult2	nein								
Add	nein	-	-	-	-			-	-
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1				-	Div

Scoreboarding: Takt +6

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**4**	
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**	**+**			

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	ja	Mult	F0	F2	F4			nein	nein
Mult2	nein								
Add	ja	Add	F6	F8	F2			ja	ja
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1			Add		Div

Scoreboarding: Takt +7

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**+**	
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**	**+**	**+**		

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	ja	Mult	F0	F2	F4			nein	nein
Mult2	nein								
Add	ja	Add	F6	F8	F2			nein	nein
Divide	ja	Div	F10	F0	F6	Mult1		nein	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	Mult1			Add		Div

Scoreboarding: Takt +8

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**+**	**+**
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**			
ADD.D	**F6,F8,F2**	**+**	**+**	**1**	

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	nein	-	-	-	-			-	-
Mult2	nein								
Add	ja	Add	F6	F8	F2			nein	nein
Divide	ja	Div	F10	F0	F6	-		ja	ja

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	-			Add		Div

Scoreboarding: Takt +9

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**+**	**+**
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**	**+**		
ADD.D	**F6,F8,F2**	**+**	**+**	**+**	

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	nein	-	-	-	-			-	-
Mult2	nein								
Add	ja	Add	F6	F8	F2			nein	nein
Divide	ja	Div	F10	F0	F6			nein	nein

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit	-			Add		Div

Scoreboarding: Takt +10

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**+**	**+**
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**	**+**	**1**	
ADD.D	**F6,F8,F2**	**+**	**+**	**+**	**+**

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	nein								
Mult2	nein								
Add	nein	-	-	-	-			-	-
Divide	ja	Div	F10	F0	F6			nein	nein

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit				-		Div

Scoreboarding: Takt +11

Befehls-Status

Befehl		*Issue*	*Read Op.*		*Exec. complete*	*Write*	
L.D	**F6,32(R2)**		+	+	+		+
L.D	**F2,96(R3)**		+	+	+		+
MUL.D	**F0,F2,F4**	+	+		+	+	
SUB.D	**F8,F6,F2**	+	+		+	+	
DIV.D	**F10,F0,F6**	+	+		**2**		
ADD.D	**F6,F8,F2**	+	+		+	+	

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	nein								
Mult2	nein								
Add	nein								
Divide	ja	Div	F10	F0	F6			nein	nein

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit						Div

Scoreboarding: folgende Takte *(nur DIV wird ausgeführt)*

Befehls-Status

Befehl		*Issue*	*Read Op.*	*Exec. complete*	*Write*
L.D	**F6,32(R2)**	**+**	**+**	**+**	**+**
L.D	**F2,96(R3)**	**+**	**+**	**+**	**+**
MUL.D	**F0,F2,F4**	**+**	**+**	**+**	**+**
SUB.D	**F8,F6,F2**	**+**	**+**	**+**	**+**
DIV.D	**F10,F0,F6**	**+**	**+**	**+**	
ADD.D	**F6,F8,F2**	**+**	**+**	**+**	**+**

Status der Funktionseinheiten

Name	*Busy*	*Op*	*Fi*	*Fj*	*Fk*	*Qj*	*Qk*	*Rj*	*Rk*
Integer	nein								
Mult1	nein								
Mult2	nein								
Add	nein								
Divide	ja	Div	F10	F0	F6			nein	nein

Register-Ergebnis-Status

	F0	*F2*	*F4*	*F6*	*F8*	*F10*
Einheit						Div

Tomasulo-Algorithmus

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

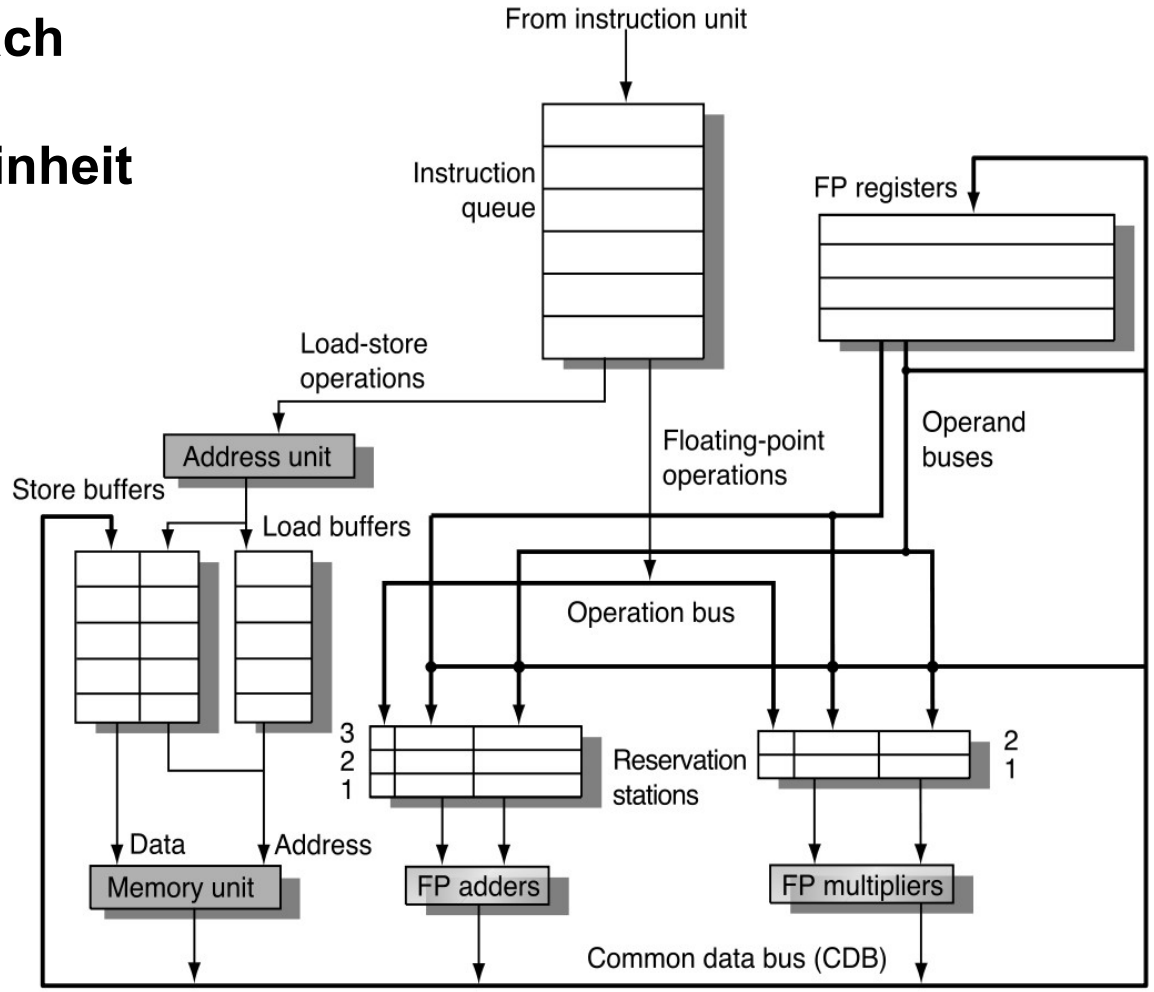
- **Falls |*Reservation Stations*| > |Register| ☞ Namenskonflikte lösbar, die Compiler nicht behandeln kann!**

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**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
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**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi	Mult1	Load2		Add2	Add3	Mult2	

Tomasulo-Algorithmus: Takt +1

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
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**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi	Mult1	-		Add2	Add3	Mult2	

Tomasulo-Algorithmus: Takt +2

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
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**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi	Mult1			Add2	Add3	Mult2	

Tomasulo-Algorithmus: Takt +3

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
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**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi	Mult1			Add2	Add3	Mult2	

Tomasulo-Algorithmus: Takt +4

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
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**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi	Mult1			Add2	Add3	Mult2	

Tomasulo-Algorithmus: Takt +5

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

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

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi	-				Add3	Mult2	

Tomasulo-Algorithmus: Takt +6

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

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

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi					Add3	Mult2	

Tomasulo-Algorithmus: Takt +7

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

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

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi					-	Mult2	

Tomasulo-Algorithmus: Takt +8

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

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

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi						Mult2	

Tomasulo-Algorithmus: Takt +9

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

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

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi						Mult2	

Tomasulo-Algorithmus: Takt +10

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

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

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi						Mult2	

Tomasulo-Algorithmus: Takt +11

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	ja	DIV					

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi						Mult2	

Tomasulo-Algorithmus: Takt +12

- **Befehls-Status**

Befehl		*Issue*	*Execute*	*Write Result*
`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**

Name	*Busy*	*Op*	*Vj*	*Vk*	*Qj*	*Qk*	*A*
Load1	nein						
Load2	nein						
Add1	nein						
Add2	nein						
Add3	nein						
Mult1	nein						
Mult2	nein						

- **Register-Ergebnis-Status**

	F0	*F2*	*F4*	*F6*	*F8*	*F10*	*...*
Qi						-	

Unterschiede Scoreboarding / Tomasulo

Scoreboarding	Tomasulo
Zentrale Instanz	Konfliktdetektion und Ausführungskontrolle verteilt
4 Stufen: issue, read operands, execution, write result	3 Stufen: issue, execution, write result
Arbeitet ausschließlich auf phys. Registern	Arbeitet auf logischen Registern (register renaming)
In jedem Takt wird zentral erfasst, welche Operanden bekannt sind	Ausstehende Operanden verweisen auf dezentral verteilte reservation stations
Antidaten- (WAR) und Ausgabeabhängigkeiten (WAW) immer durch Stalls gelöst	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