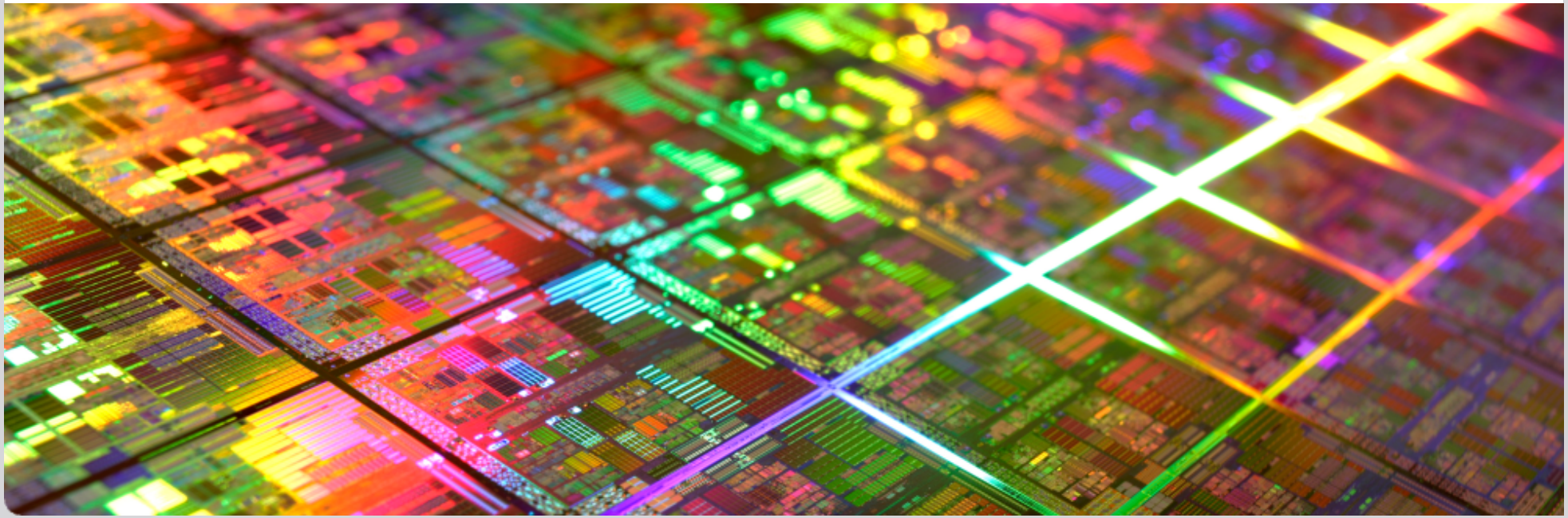
Karlsruher Institut für Technologie

Zentralübung Rechnerstrukturen im SS 2015
Pipelining und Superskalartechniken

Mario Kicherer, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

26. Mai 2015



Pipelining

Aufgabe 1

Die Befehlsabarbeitung eines Prozessors lässt sich in folgende Phasen einteilen:

IF	ID	EX	MA	WB
250 ps	100 ps	130 ps	220 ps	50 ps

Wenn der Prozessor mit einer 5-stufigen Pipeline (für jede Phase eine Pipelinestufe) ausgestattet wird, kommt bedingt durch das nötige Pipelineregister eine weitere Latenz von 20 ps hinzu.

Pipelining - Aufgabe 1

IF	ID	EX	MA	WB
250 ps	100 ps	130 ps	220 ps	50 ps

a) **Wie groß ist die Zykluszeit des Prozessors ohne Pipeline, wie groß mit der 5-stufigen Pipeline?**

- Ohne Pipelining:
  Zykluszeit = Summe aller Stufen
  Zykluszeit = 250 ps + 100 ps + 130 ps + 220 ps + 50 ps = 750 ps
- Mit Pipelining:
  Zykluszeit = Längste Stufe + Latenz des Pipelineregisters
  Zykluszeit = 250 ps (IF-Stufe) + 20 ps = 270 ps

Pipelining - Aufgabe 1

IF	ID	EX	MA	WB
250 ps	100 ps	130 ps	220 ps	50 ps

b) **Welche Beschleunigung (Speed-Up) ergibt sich unter der Annahme, dass der Prozessor ohne Pipeline einen CPI von 1,0 und die Version mit Pipeline einen CPI von 1,2 hat?**

$$\begin{aligned} SpeedUp &= \frac{\text{average exec time w/o pipeline}}{\text{average exec time w pipeline}} \\ &= \frac{\text{CPI * CycleTime w/o p}}{\text{CPI * CycleTime w p}} \\ &= \frac{1,0 * 750\,ps}{1,2 * 270\,ps} \approx 2,31 \end{aligned}$$

Pipelining - Aufgabe 1

IF	ID	EX	MA	WB
250 ps	100 ps	130 ps	220 ps	50 ps

c) **Wenn der Prozessor mit einer 3-stufigen Pipeline implementiert werden soll, müssen die existierenden 5 Pipelinestufen kombiniert werden. Wie würden Sie die 5 Phasen auf die drei Pipelinestufen aufteilen, um eine möglichst kurze Zykluszeit zu erhalten?**

IF	ID + EX	MA + WB

- 1. IF
- 2. ID + EX
- 3. MA + WB

Pipelining - Aufgabe 1

IF	ID	EX	MA	WB
250 ps	100 ps	130 ps	220 ps	50 ps

d) **Wenn der Prozessor auch mit einer 6-stufigen Pipeline realisiert werden könnte, welche Pipelinestufe würden Sie aufteilen, um die Zykluszeit des Prozessors zu senken?**

- Die Zykluszeit wird von der Ausführungszeit längsten Stufe bestimmt

$\Rightarrow$ Aufteilung der längsten Stufe, hier der IF-Stufe

- Die Aufteilung einer anderen Stufe würde zu keiner Reduktion der Zykluszeit führen.

Superskalartechnik

Überblick

- Dynamische Parallelisierung eines sequentiellen Befehlsstroms
- Pro Takt werden ein oder mehrere Befehle aus dem Programmspeicher geholt und dekodiert
- Pro Takt können mehrere Befehle den Ausführungseinheiten zugewiesen werden
- **Konfliktauflösung:** Parallele Ausführbarkeit der Befehle wird durch die Hardware ermittelt
- **Dynamisches Konzept**
- Weit verbreitet
- **Sequentielles Erscheinungsbild**

Superskalare Pipelines – Stufen

Ablauf: 5 Phasen (nach Brinkschulte / Ungerer)

- **Dekodierung**: Erkennen der Befehlsklasse, der Adressierungsmodi usw.
- **Registerumbenennung**: Belegen eines physischen Registers mit dem virtuellen
- **Befehlszuordnung**: Eintragung in die Reservation Station - Zuordnung eines Befehls zu einer Einheitenfamilie (**Issue**) und Anstoßen bei Verfügbarkeit der Operandenwerte (**Dispatch**)
- **Ausführung**: Ausführung der Rechenoperation oder des Speicherzugriffs
- **Rückordnung**: Schreiben des physischen Registers

Dekodierung und **Registerumbenennung** können dabei zusammengefasst werden.

Algorithmus von Tomasulo

Tomasulo

- Anstoßen der Befehle **dezentral** aus den Reservation Stations

Empty	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2

- Einlesen der Operanden **dezentral** in die Reservation Stations
- Übertragung der Operandenwerte über den **Common Data Bus** (CDB): Tupel aus Tag/Wert
- Result Forwarding ebenfalls **dezentral**
- Einheiten können zu **Einheitenfamilien** zusammengefasst werden mit einer gemeinsamen, mehrzeiligen Reservation Station Table
- Reorder Buffer für die Befehlsrückordnung (Wiederherstellung der sequentiellen Programmsemantik)

Bestandteile einer Implementierung

Befehlsfenster

Nummer	Befehl	Stufe
1	`Op Operands`	–

- Zuständig für die Bereitstellung dekodierter Befehle an Zuweisungseinheit (**Instruction Issue**)
- Begrenzte Größe
- Die hier gespeicherten Befehle sind frei von Namens-Abhängigkeiten und Steuerflussabhängigkeiten
- Hier erweitert um Feld der derzeitigen Stufe

Bestandteile einer Implementierung

Registerstatustabelle

Feld	R0	R1	R2	F0	F2	...
Value	42	23	-	47,11	0	...
Valid	1	1	0	1	1	
RS	0	0	1	0	0	

- Gibt bei Verwendung von Register-Renaming auch Auskunft über Abbildung von Architekturregistern auf physische Register

Bestandteile einer Implementierung

Register-Renaming – Beispiel

```
add   r4, r2, r1
div   r2, r3, r4
store (r5), r2
add   r2, r1, r4
```

(Ausgabe-) Abhängigkeit zwischen Befehl 2 und 4

$\Rightarrow$

```
add   r4, r2, r1
div   r2, r3, r4
store (r5), r2
add   r6, r1, r4
```

Nach Registerumbenennung: keine Ausgabe- oder Gegen-Abhängigkeit mehr vorhanden

$\Rightarrow$ Befehle 2 und 4 können parallel ausgeführt werden

Bestandteile einer Implementierung

Reservation Stations (Reservierungstabelle)

Unit	Emp.	InFU	Op	Dest	Src1	Vld 1	RS1	Src2	Vld2	RS2
L/S 1 L/S 2										
Int-Add Int-Mul										
FP-Add										

- Bei Tomasulo dezentral an jeder Einheit oder Familie von Einheiten
- Jede Ausführungseinheit kann über mehrere Einträge verfügen
- **Heute:** Tendenz zur zentralen Zuteilung
  Bsp.: Intel Westmere - 36 Entry Unified Scheduler

Bestandteile einer Implementierung

Rückordnungspuffer

Befehlsnr.	Ziel	Quelle
2	Phys. Reg.	Einheit
1	Phys. Reg.	Einheit

- Wird bei Eintragung eines Befehls in Reservation Station gefüllt und enthält Zielregister, Befehlsnummer sowie produzierende Einheit
- Wird von **Retirement Unit** (Rückordnungsstufe) benutzt, um **Commitment** oder **Removement** durchzuführen

$\Rightarrow$ Rückordnung der Befehle in Programmreihenfolge

Tomasulo – Aufgabe 2

Karlsruher Institut für Technologie

Aufgabenstellung

Unter Verwendung des Algorithmus von Tomasulo, bestimmen Sie für jede Instruktion in der unten stehenden Sequenz, in welcher sie zugeteilt (**Issue**), sie zur Ausführung angestoßen (**Execute**) und wann ihr entsprechendes Ergebnis auf den Ergebnisbus gelegt wird (**Write Result**).
Beginnen Sie mit Takt 1.

Tomasulo – Aufgabe 2

Annahmen - Auszug

- Die **Multiplikation** benötigt **vier** Takte, die **Division acht** Takte, alle **anderen Operationen zwei** Takte zur Ausführung.
- Der Prozessor verfüge über **eine Mul-/Div**-Einheit und **eine Add-/Sub**-Einheit.
- Die **Mul-/Div**-Einheit hat eine Reservierungstabelle mit **zwei** Einträgen, die der **Add-/Sub**-Einheit **vier**.
- Der Prozessor kann nur **eine Instruktion** pro Takt auf die Reservierungstabellen **zuteilen**.
- Die Instruktionen werden **in-order** zugeteilt.
- Der Prozessor verfüge über nur **einen Ergebnisbus**.
- Wenn mehrere Operationen gleichzeitig fertig werden, hat die **Add-/Sub-Einheit Vorrang** gegenüber der Mul-/Div-Einheit.

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`			
2	`div r4, r4, r2`			
3	`add r1, r4, r4`			
4	`add r2, r4, r3`			
5	`div r1, r2, r3`			
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
0

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt 0												
Add/Sub	0												
Mul/Div	0												

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1		
2	`div r4, r4, r2`			
3	`add r1, r4, r4`			
4	`add r2, r4, r3`			
5	`div r1, r2, r3`			
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
1

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1											
Add/Sub	0	0											
Mul/Div	0	1											

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	
2	`div r4, r4, r2`	2		
3	`add r1, r4, r4`			
4	`add r2, r4, r3`			
5	`div r1, r2, r3`			
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
2

Befehl 1 ausführen, Mul/Div-RS voll

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2										
Add/Sub	0	0	0										
Mul/Div	0	1	2										

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	
2	`div r4, r4, r2`	2		
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`			
5	`div r1, r2, r3`			
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
3

Zuteilung Befehl 3,
Befehl 2 wartet auf 1

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3									
Add/Sub	0	0	0	1									
Mul/Div	0	1	2	2									

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	
2	`div r4, r4, r2`	2		
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`			
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
4

Zuteilung Befehl 4,
Befehl 3 wartet auf 2

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4								
Add/Sub	0	0	0	1	2								
Mul/Div	0	1	2	2	2								

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2		
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`			
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
6

Ergebnis Befehl 1 auf Bus

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4								
Add/Sub	0	0	0	1	2								
Mul/Div	0	1	2	2	2								

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`			
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
7

Befehl 2 startet,
Platz für Befehl 5

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4								
Add/Sub	0	0	0	1	2								
Mul/Div	0	1	2	2	2								

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`			
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
8

Befehl 5 wartet auf Befehl 4, Zuteilung Befehl 6

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8							
Add/Sub	0	0	0	1	2	3							
Mul/Div	0	1	2	2	2	2							

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
9

Befehl 6 wartet auf Befehl 4,
Zuteilung Befehl 7, RS voll

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3		
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`			
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
15

Befehl 2 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
16

Start Befehl 3,
Zuteilung Befehl 8

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4		
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`			
10	`sub r4, r4, r1`			

Cycle
18

Befehl 3 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`			

Cycle
19

Start Befehl 4,
Zuteilung Befehl 9

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7		
6	`sub r4, r4, r2`	8		
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`			

Cycle
21

Befehl 4 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	
6	`sub r4, r4, r2`	8	22	
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`	22		

Cycle
22

Start Befehl 5 und 6, Zuteilung Befehl 10

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`	22		

Cycle
24

Befehl 6 fertig,
Befehl 7 wartet auf 5

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9						
Add/Sub	0	0	0	1	2	3	4						
Mul/Div	0	1	2	2	2	2	2						

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`	22		

Cycle
25

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25					
Add/Sub	0	0	0	1	2	3	4	3					
Mul/Div	0	1	2	2	2	2	2	2					

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9		
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`	22		

Cycle
30

Befehl 5 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25					
Add/Sub	0	0	0	1	2	3	4	3					
Mul/Div	0	1	2	2	2	2	2	2					

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`	22		

Cycle
31

Start Befehl 7

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31				
Add/Sub	0	0	0	1	2	3	4	3	3				
Mul/Div	0	1	2	2	2	2	2	2	1				

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16		
9	`add r3, r3, r3`	19		
10	`sub r4, r4, r1`	22		

Cycle
33

Befehl 7 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31				
Add/Sub	0	0	0	1	2	3	4	3	3				
Mul/Div	0	1	2	2	2	2	2	2	1				

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	
9	`add r3, r3, r3`	19	34	
10	`sub r4, r4, r1`	22		

Cycle
34

Start Befehl 8 und 9

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34			
Add/Sub	0	0	0	1	2	3	4	3	3	2			
Mul/Div	0	1	2	2	2	2	2	2	1	1			

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	
9	`add r3, r3, r3`	19	34	36
10	`sub r4, r4, r1`	22		

Cycle
36

Befehl 9 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34			
Add/Sub	0	0	0	1	2	3	4	3	3	2			
Mul/Div	0	1	2	2	2	2	2	2	1	1			

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	
9	`add r3, r3, r3`	19	34	36
10	`sub r4, r4, r1`	22		

Cycle
37

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34	37		
Add/Sub	0	0	0	1	2	3	4	3	3	2	1		
Mul/Div	0	1	2	2	2	2	2	2	1	1	1		

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	38
9	`add r3, r3, r3`	19	34	36
10	`sub r4, r4, r1`	22		

Cycle
38

Befehl 8 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34	37		
Add/Sub	0	0	0	1	2	3	4	3	3	2	1		
Mul/Div	0	1	2	2	2	2	2	2	1	1	1		

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	38
9	`add r3, r3, r3`	19	34	36
10	`sub r4, r4, r1`	22	39	

Cycle
39

Start Befehl 10

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34	37	39	
Add/Sub	0	0	0	1	2	3	4	3	3	2	1	1	
Mul/Div	0	1	2	2	2	2	2	2	1	1	1	0	

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	38
9	`add r3, r3, r3`	19	34	36
10	`sub r4, r4, r1`	22	39	41

Cycle
41

Befehl 10 fertig

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34	37	39	
Add/Sub	0	0	0	1	2	3	4	3	3	2	1	1	
Mul/Div	0	1	2	2	2	2	2	2	1	1	1	0	

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	`mul r2, r1, r1`	1	2	6
2	`div r4, r4, r2`	2	7	15
3	`add r1, r4, r4`	3	16	18
4	`add r2, r4, r3`	4	19	21
5	`div r1, r2, r3`	7	22	30
6	`sub r4, r4, r2`	8	22	24
7	`add r3, r1, r2`	9	31	33
8	`mul r1, r2, r3`	16	34	38
9	`add r3, r3, r3`	19	34	36
10	`sub r4, r4, r1`	22	39	41

Cycle
42

Füllstand der Reservation Stations bei Änderungen im jeweiligen Takt:

Unit	Takt												
	0	1	2	3	4	8	9	25	31	34	37	39	42
Add/Sub	0	0	0	1	2	3	4	3	3	2	1	1	0
Mul/Div	0	1	2	2	2	2	2	2	1	1	1	0	0

Tomasulo – Aufgabe 3

Annahmen & Voraussetzungen

- Zwei Lade-Speichereinheiten (Load/Store Unit), eine Integer-Additionseinheit, eine Integer-Multiplikationseinheit, eine FP-Additionseinheit
- Statische Sprungvorhersage mit fortwährendem Füllen der Pipeline vom Sprungziel; dafür sei ein Sprungzieladresscache vorhanden und die Sprungvorhersage laute auf Taken
- FP-Register und normale Register können gleichzeitig in der WB-Stufe beschrieben werden
- Schreiben in Speicher geschehe nicht über CDB

Tomasulo – Aufgabe 3

Annahmen & Voraussetzungen

- Die **Befehlszuordnungs- und Rückordnungs**bandbreite betrage **4** Befehle; **zwei** Befehle werden pro Takt maximal **geholt**
- Entsprechend gibt es **zwei Dekodiereinheiten**, die gleichzeitig arbeiten können
- Die Auswertung der Sprungzieladresse erfolge in der Stufe Execute der Int-Add-Einheit; das Schreiben des Befehlszählers (Instruction Counter, Program Counter) in der WB-Stufe
- Speicherlesezugriffe erfolgen analog zu normalen Rechenoperationen in der Ausführungsstufe, das Rückschreiben geschieht dabei als separater Schritt (WB)
- Speicherschreibzugriffe haben eine Ausführungsdauer von 3 Takten

Tomasulo – Aufgabe 3

Ausführungseinheiten

Folgende Pipelinestruktur: IF ID IS EX/MA (WB)

Einheit	**L/S**	**Int-Add/Sub**	**Int-Mul**	**FP-Add**
Anzahl	2	1	1	1
Bearbeitungsdauer (in Takten)	3	1	3	2

Tomasulo – Aufgabe 3

Auszuführender Programmcode

```
LOOP: LD.D   F0,0(R1)    ; loads Mem[i]
      ADD.D  F4,F0,F2    ; adds to Mem[i]
      S.D    0(R1),F4    ; stores into Mem[i]
      ADD    R1,R1,#8    ; R1 = R1+8
      SUB    R3,R1,R2    ; R3 = R1-R2
      BLTZ   R3,LOOP     ; branch if R1 < R2
```

- `R1` enthält eine Speicheradresse, `F2` fest.
- Die Schleife wird drei mal durchlaufen wegen `R2 = R1 + 24`

Tomasulo – Aufgabe 3

Takt 1

- `LD.D` und `ADD.D` geholt

Takt 2

- `LD.D` und `ADD.D` dekodiert
- `SD.D` und `ADD` geholt
- Reservation Station noch leer

Code

```
LOOP: LD.D   F0,0(R1)
      ADD.D  F4,F0,F2
      S.D    0(R1),F4
      ADD    R1,R1,#8
      SUB    R3,R1,R2
      BLTZ   R3,LOOP
```

Pipeline:

Befehl	Stufe 1	Stufe 2
`LD.D F0,0(R1)`	IF	ID
`ADD.D F4,F0,F2`	IF	ID
`S.D 0(R1),F4`		IF
`ADD R1,R1,#8`		IF

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
-		

Registerstatustabelle:

Feld	R1	R2	R3	F0	F2	F4
Value	(R1)	(R2)	(R3)	-	(F2)	(F4)
RS						

Tomasulo – Aufgabe 3

Befehlsfenster nach Takt 1 & 2:

Nummer	Befehl	Stage
1	`L.D F0,0(R1)`	ID
2	`ADD.D F4,F0,F2`	ID
-	-	-

Tomasulo – Aufgabe 3

Takt 3 & 4

- `SUB` und `BLTZ` geholt, dekodiert
- `LD.D` und `ADD.D` geholt

Befehlsfenster:

Nummer	Befehl		Stage
1	`LD.D`	`F0,0(R1)`	M
2	`ADD.D`	`F4,F0,F2`	IS
3	`S.D`	`0(R1),F4`	IS
4	`ADD`	`R1,R1,#8`	IS
5	`SUB`	`R3,R1,R2`	ID
6	`BLTZ`	`R3,LOOP`	ID

Code

```
LOOP: LD.D   F0,0(R1)
      ADD.D  F4,F0,F2
      S.D    0(R1),F4
      ADD    R1,R1,#8
      SUB    R3,R1,R2
      BLTZ   R3,LOOP
```

Pipeline:

Befehl		Takt 1	2	3	4
`LD.D`	`F0,0(R1)`	IF	ID	IS	M
`ADD.D`	`F4,F0,F2`	IF	ID	IS	
`S.D`	`0(R1),F4`		IF	ID	IS
`ADD`	`R1,R1,#8`		IF	ID	IS
`SUB`	`R3,R1,R2`			IF	ID
`BLTZ`	`R3,LOOP`			IF	ID
`LD.D`	`F0,0(R1)`				IF
`ADD.D`	`F4,F0,F2`				IF

Tomasulo – Aufgabe 3

Karlsruher Institut für Technologie

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
4	R1	Int-Add
3	null	any L/S
2	F4	FP-Add
1	F0	L/S 1

Reservation Stations:

Unit	Em.	InFU	Op	Dest	Src1	Vld1	RS1	Src2	Vld2	RS2
L/S 1	0	1	ld.d	F0	[(R1)]	1	0	-	-	-
L/S 2	0	0	s.d	(M)	R1	1	0	-	0	FP-Add
Int-A/S	0	0	add	R1	(R1)	1	0	8	1	-
Int-Mul	1									
FP-Add	0	0	add.d	F4	-	0	L/S 1	(F2)	1	-

Registerstatustabelle:

Feld	R1	R2	R3	F0	F2	F4
Value	-	(R2)	(R3)	-	(F2)	-
RS	Int-A/S			L/S 1		FP-Add

Tomasulo – Aufgabe 3

Takt 5 & 6

- `ADD` wird ausgeführt und schreibt `R1`
- Zuordnung von `SUB` noch im gleichen Takt

Befehlsfenster:

Nummer	Befehl		Stage
1	`LD.D`	`F0,0(R1)`	M
2	`ADD.D`	`F4,F0,F2`	IS
3	`S.D`	`0(R1),F4`	IS
4	`ADD`	`R1,R1,#8`	WB
5	`SUB`	`R3,R1,R2`	IS
6	`BLTZ`	`R3,LOOP`	ID
7	`LD.D`	`F0,0(R1)`	ID
8	`ADD.D`	`F4,F0,F2`	ID
9	`S.D`	`0(R1),F4`	ID
10	`ADD`	`R1,R1,#8`	ID

Pipeline:

Befehl		Takt			
		3	4	5	6
`LD.D`	`F0,0(R1)`	IS	M	M	M
`ADD.D`	`F4,F0,F2`	IS			
`S.D`	`0(R1),F4`	ID	IS		
`ADD`	`R1,R1,#8`	ID	IS	EX	WB
`SUB`	`R3,R1,R2`	IF	ID		IS
`BLTZ`	`R3,LOOP`	IF	ID		
`LD.D`	`F0,0(R1)`		IF	ID	
`ADD.D`	`F4,F0,F2`		IF	ID	
`S.D`	`0(R1),F4`			IF	ID
`ADD`	`R1,R1,#8`			IF	ID
`SUB`	`R3,R1,R2`				IF
`BLTZ`	`R3,LOOP`				IF

Tomasulo – Aufgabe 3

Karlsruher Institut für Technologie

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
5	R3	Int-Add
4	**R1**	**Int-Add**
3	null	any L/S Unit
2	F4	FP-Add
1	F0	L/S 1

Reservation Stations:

Unit	Em.	InFU	Op	Dest	Src1	Vld1	RS1	Src2	Vld2	RS2
L/S 1	0	1	ld.d	F0	[(R1)]	1	0	-	-	-
L/S 2	0	0	s.d	(M)	R1	1	0	-	0	FP-Add
Int-A/S	0	0	sub	R3	(R1)+8	1	-	(R2)	1	-
Int-Mul	1									
FP-Add	0	0	add.d	F4	-	0	L/S 1	(F2)	1	-

Registerstatustabelle:

Feld	R1	R2	R3	F0	F2	F4
Value	(R1)+8	(R2)	-	-	(F2)	-
RS			Int-A/S	L/S 1		FP-Add

Tomasulo – Aufgabe 3

Takt 7 & 8

- Erstes `LD.D` beendet, Weiterleitung an FP-ADD
- `ADD.D` Ausführung begonnen
- `SUB` angestoßen und berechnet
- `BLTZ` zugeteilt
- Zweites `LD.D` beginnt Ausführung

Pipeline:

Befehl		**Takt**					
		3	4	5	6	7	8
`LD.D`	`F0,0(R1)`	IS	M	M	M	WB	
`ADD.D`	`F4,F0,F2`	IS				EX	EX
`S.D`	`0(R1),F4`	ID	IS				
`ADD`	`R1,R1,#8`	ID	IS	EX	WB		
`SUB`	`R3,R1,R2`	IF	ID	ID	IS	EX	WB
`BLTZ`	`R3,LOOP`	IF	ID				IS
`LD.D`	`F0,0(R1)`		IF	ID		IS	M
`ADD.D`	`F4,F0,F2`		IF	ID			
`S.D`	`0(R1),F4`			IF	ID		
`ADD`	`R1,R1,#8`			IF	ID		
`SUB`	`R3,R1,R2`				IF	ID	
`BLTZ`	`R3,LOOP`				IF	ID	
`LD.D`	`F0,0(R1)`					IF	ID
`ADD.D`	`F4,F0,F2`					IF	ID
`S.D`	`0(R1),F4`						IF
`ADD`	`R1,R1,#8`						IF

Tomasulo – Aufgabe 3

Befehlsfenster:

Nummer	Befehl		Stage
2	ADD.D	F4,F0,F2	EX
3	S.D	0(R1),F4	IS
5	SUB	R3,R1,R2	WB
6	BLTZ	R3,LOOP	IS
7	LD.D	F0,0(R1)	M
8	ADD.D	F4,F0,F2	ID
9	S.D	0(R1),F4	ID
10	ADD	R1,R1,#8	ID
11	SUB	R3,R1,R2	ID
12	BLTZ	R3,LOOP	ID
13	LD.D	F0,0(R1)	ID
14	ADD.D	F4,F0,F2	ID

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
7	F0	L/S 2
6	PC	Int-Add
5	**R3**	**Int-Add**
4	**R1**	**Int-Add**
3	null	any L/S Unit
2	F4	FP-Add

Die vollendeten Add- und Subbefehle können noch nicht zurückgeordnet werden, da die vorherigen Befehle noch nicht alle beendet sind.

Tomasulo – Aufgabe 3

Karlsruher Institut für Technologie

Reservation Stations:

Unit	Em.	InFU	Op	Dest	Src1	Vld1	RS1	Src2	Vld2	RS2
L/S 1	0	1	ld.d	F0	(R1)+8	1	0	-	-	-
L/S 2	0	0	s.d	(M)	R1	1	0	-	0	FP-Add
Int-A/S	0	0	bltz	PC	(R1-R2)	1	0	-	-	-
Int-Mul	1									
FP-Add	0	1	add.d	F4	(F0)	1	-	(F2)	1	-

Registerstatustabelle:

Feld	**R1**	**R2**	**R3**	**F0**	**F2**	**F4**
Value	(R1)+8	(R2)	(R1-R2)	-	(F2)	-
RS				L/S 1		FP-Add

Tomasulo – Aufgabe 3

Befehl		Takt											
		1	2	3	4	5	6	7	8	9	10	11	12
LD.D	F0,0(R1)	IF	ID	IS	M	M	M	WB					
ADD.D	F4,F0,F2	IF	ID	IS				EX	EX	WB			
S.D	0(R1),F4		IF	ID	IS					M	M	M	
ADD	R1,R1,#8		IF	ID	IS	EX	WB						
SUB	R3,R1,R2			IF	ID		IS	EX	WB				
BLTZ	R3,LOOP			IF	ID				IS	EX	WB		
LD.D	F0,0(R1)				IF	ID		IS	M	M	M	WB	
ADD.D	F4,F0,F2				IF	ID				IS		EX	EX
S.D	0(R1),F4					IF	ID					IS	
ADD	R1,R1,#8					IF	ID				IS	EX	WB
SUB	R3,R1,R2						IF	ID					IS
BLTZ	R3,LOOP						IF	ID					
LD.D	F0,0(R1)							IF	ID			IS	
ADD.D	F4,F0,F2							IF	ID				
S.D	0(R1),F4								IF	ID			
ADD	R1,R1,#8								IF	ID			
SUB	R3,R1,R2									IF	ID		
BLTZ	R3,LOOP									IF	ID		

Tomasulo – Aufgabe 3

Befehl		Takt									
		13	14	15	16	17	18	19	20	21	22
LD.D	F0,0(R1)										
ADD.D	F4,F0,F2										
S.D	0(R1),F4										
ADD	R1,R1,#8										
SUB	R3,R1,R2										
BLTZ	R3,LOOP										
LD.D	F0,0(R1)										
ADD.D	F4,F0,F2	WB									
S.D	0(R1),F4	M	M	M							
ADD	R1,R1,#8										
SUB	R3,R1,R2	EX	WB								
BLTZ	R3,LOOP		IS	EX	WB						
LD.D	F0,0(R1)	M	M	M	WB						
ADD.D	F4,F0,F2	IS			EX	EX	WB				
S.D	0(R1),F4			IS			M	M	M		
ADD	R1,R1,#8				IS	EX	WB				
SUB	R3,R1,R2						IS	EX	WB		
BLTZ	R3,LOOP								IS	EX	WB

VLIW-Prozessoren

Überblick

- Befehlswort aus mehreren einzelnen Befehlen zusammengesetzt
- Parallelität **explizit vom Compiler** angegeben
- **Statisches** Konzept
- „Platzhalter“ in Befehlswort für jede vorhandene Ausführungseinheit
- Sinnvoll bei Spezialanwendungen: DSP, Graphik, Netzwerk
- Moderne Varianten: **EPIC** (Intel Itanium), Transmeta Crusoe

Befehlsformat

Befehl 1	Befehl 2	Befehl 3	Befehl 4	Steuerbits

VLIW-Prozessoren – Aufgabe 4

Karlsruher Institut für Technologie

Der folgende Assembler-Code soll auf einem VLIW-Prozessor mit drei parallelen Ausführungseinheiten ausgeführt werden. Geben Sie hierfür eine möglichst effiziente Befehlsverteilung an. Die Befehle können beliebig umsortiert werden, so lange die Korrektheit der Anwendung gewährleistet ist.

```
1   add     r1, r2, r3          ; r1 = r2 + r3
2   sub     r5, r3, r5          ; r5 = r3 - r5
3   ld      r3, [r1]            ; Load r3 with [r1]
4   mul     r3, r3, r3          ; r3 = r3 * r3
5   st      [r5], r3            ; Store r3 in [r5]
6   ld      r9, [r7]            ; Load r9 with [r7]
7   ld      r11, [r12]          ; Load r11 with [r12]
8   add     r11, r11, r12       ; r11 = r11 + r12
9   mul     r11, r11, r9        ; r11 = r11 * r9
10   st     [r12], r11          ; Store r11 in [r12]
```

VLIW-Prozessoren – Aufgabe 4

Karlsruher Institut für Technologie

Aufgabe 4a)

Nehmen sie vereinfachend an, dass alle Befehle innerhalb eines Taktes abgearbeitet werden können.

Aufgabe 4b)

Nehmen Sie nun an, dass die ersten beiden Ausführungseinheiten arithmetisch-logische Befehle (add, sub, mul) ausführen können und die dritte für Load-/Store Operationen zuständig ist.

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Initialer Zustand

Slot 1	Slot 2	Slot 3

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 1

Slot 1	Slot 2	Slot 3
add r1, r2, r3		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 2

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 3
Abhängigkeiten beachten

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	
ld r3, [r1]		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 4
Abhängigkeiten beachten

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	
ld r3, [r1]		
mul r3, r3, r3		

VLIW-Prozessoren – Aufgabe 4a)

```
 1   add     r1, r2, r3
 2   sub     r5, r3, r5
 3   ld      r3, [r1]
 4   mul     r3, r3, r3
 5   st      [r5], r3
 6   ld      r9, [r7]
 7   ld      r11, [r12]
 8   add     r11, r11, r12
 9   mul     r11, r11, r9
10   st      [r12], r11
```

Zuordnung Befehl 5
Abhängigkeiten beachten

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	
ld r3, [r1]		
mul r3, r3, r3		
st [r5], r3		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 6

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	ld r9, [r7]
ld r3, [r1]		
mul r3, r3, r3		
st [r5], r3		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 7

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	ld r9, [r7]
ld r3, [r1]	ld r11, [r12]	
mul r3, r3, r3		
st [r5], r3		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 8

- ⇒ Befehle 9 und 10 auch von Befehl 7 abhängig
- ⇒ führt zu langer Befehlsfolge und einem nötigen 5. Befehl
- ⇒ Optimierung notwendig

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	ld r9, [r7]
ld r3, [r1]	ld r11, [r12]	
mul r3, r3, r3	add r11, r11, r12	
st [r5], r3		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Vertauschen von Befehl 6 und 7 und Neuordnung von Befehl 8

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	ld r11, [r12]
ld r3, [r1]	ld r9, [r7]	add r11, r11, r12
mul r3, r3, r3		
st [r5], r3		

VLIW-Prozessoren – Aufgabe 4a)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 9

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	ld r11, [r12]
ld r3, [r1]	ld r9, [r7]	add r11, r11, r12
mul r3, r3, r3	mul r11, r11, r9	
st [r5], r3		

VLIW-Prozessoren – Aufgabe 4a)

```
 1   add     r1, r2, r3
 2   sub     r5, r3, r5
 3   ld      r3, [r1]
 4   mul     r3, r3, r3
 5   st      [r5], r3
 6   ld      r9, [r7]
 7   ld      r11, [r12]
 8   add     r11, r11, r12
 9   mul     r11, r11, r9
10   st      [r12], r11
```

Zuordnung Befehl 10

Slot 1	Slot 2	Slot 3
add r1, r2, r3	sub r5, r3, r5	ld r11, [r12]
ld r3, [r1]	ld r9, [r7]	add r11, r11, r12
mul r3, r3, r3	mul r11, r11, r9	
st [r5], r3	st [r12], r11	

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Initialer Zustand

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 1

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3		

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 2

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 3
Abhängigkeiten beachten

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	
		ld r3, [r1]

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 4
Abhängigkeiten beachten

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	
		ld r3, [r1]
mul r3, r3, r3		

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 5
Abhängigkeiten beachten

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	
		ld r3, [r1]
mul r3, r3, r3		
		st [r5], r3

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 6

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	ld r9, [r7]
		ld r3, [r1]
mul r3, r3, r3		
		st [r5], r3

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 7
Ressourcenkonflikt

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	ld r9, [r7]
		ld r3, [r1]
mul r3, r3, r3		ld r11, [r12]
		st [r5], r3

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 8

- ⇒ Befehle 9 und 10 auch von Befehl 7 abhängig
- ⇒ führt zu langer Befehlsfolge und weiteren nötigen Befehlen
- ⇒ Optimierung notwendig

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	ld r9, [r7]
		ld r3, [r1]
mul r3, r3, r3		ld r11, [r12]
	add r11, r11, r12	st [r5], r3

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Vertauschen von Befehl 6 und 7 und Neuordnung von Befehl 8

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	ld r11, [r12]
add r11, r11, r12		ld r3, [r1]
mul r3, r3, r3		ld r9, [r7]
		st [r5], r3

VLIW-Prozessoren – Aufgabe 4b)

```
1   add     r1, r2, r3
2   sub     r5, r3, r5
3   ld      r3, [r1]
4   mul     r3, r3, r3
5   st      [r5], r3
6   ld      r9, [r7]
7   ld      r11, [r12]
8   add     r11, r11, r12
9   mul     r11, r11, r9
10  st      [r12], r11
```

Zuordnung Befehl 9

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	ld r11, [r12]
add r11, r11, r12		ld r3, [r1]
mul r3, r3, r3		ld r9, [r7]
mul r11, r11, r9		st [r5], r3

VLIW-Prozessoren – Aufgabe 4b)

```
 1   add     r1, r2, r3
 2   sub     r5, r3, r5
 3   ld      r3, [r1]
 4   mul     r3, r3, r3
 5   st      [r5], r3
 6   ld      r9, [r7]
 7   ld      r11, [r12]
 8   add     r11, r11, r12
 9   mul     r11, r11, r9
10   st      [r12], r11
```

Zuordnung Befehl 10

Slot 1 (ALU)	Slot 2 (ALU)	Slot 3 (L/S)
add r1, r2, r3	sub r5, r3, r5	ld r11, [r12]
add r11, r11, r12		ld r3, [r1]
mul r3, r3, r3		ld r9, [r7]
mul r11, r11, r9		st [r5], r3
		st [r12], r11

Karlsruher Institut für Technologie

Zentralübung Rechnerstrukturen im SS 2015
Pipelining und Superskalartechniken

Mario Kicherer, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

26. Mai 2015