

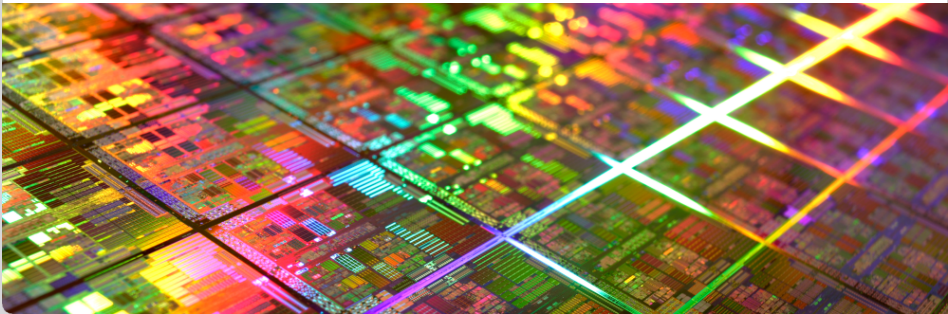
Zentralübung Rechnerstrukturen im SS 2012

Prozessorarchitektur - Pipelining und Superskalartechniken

Martin Schindewolf, Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

19. Juni 2012



- Pipelining
- Superskalartechnik: Algorithmus von Tomasulo
- VLIW-Prozessoren

Aufgabe 1

Die Befehlsabarbeitung eines Prozessors lässt sich in folgende Phasen einteilen:

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

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 20ps hinzu.

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

a)

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

Antwort

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

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

b)

Unter der Annahme, dass der Prozessor ohne Pipeline ein CPI von 1.0 und die Version mit Pipeline ein CPI von 1.2 hat: welcher Speed-Up ergibt sich?

Antwort

$$SpeedUp = \frac{\text{average exec time old}}{\text{average exec time new}} = \frac{CPI * CycleTime\ old}{CPI * CycleTime\ new} = \frac{1 * 750ps}{1.2 * 270ps} \approx 2.31$$

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

c)

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

Antwort

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

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

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?

Antwort

Zykluszeit wird von der längsten Stufe bestimmt

⇒ Aufteilung der längsten Stufe (hier IF-Stufe)

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

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

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.

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)

Befehlsfenster

Nummer	Befehl	Stufe
1	Op Operands	—

- Zuständig für die Bereitstellung dekodierter Befehle an Zuweisungseinheit (**Instruction Issue**)
- Begrenzte Größe
- Zuweisung von bis zu sechs Anweisungen pro Takt
- Die hier gespeicherten Befehle sind frei von Namens-Abhängigkeiten und Steuerflussabhängigkeiten
- Hier erweitert um Feld der derzeitigen Stufe; dadurch auch mehr Einträge als in echter Hardware

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

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

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

Nach
Registerumbenennung:
keine Ausgabe oder
Gegen-Abhängigkeit mehr
vorhanden
⇒ Befehle 2 und 4 können
parallel ausgeführt werden

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

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
- ⇒ Rückordnung der Befehle in Programmreihenfolge

Aufgabe

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**).

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:

Unit	Takt																		
	0																		
Add/Sub	0																		
Mul/Div	0																		

Tomasulo – Aufgabe 2

#	Instruktion	Issue	Execute	Write Result
1	mul r2, r1, r1	1	2	6
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:

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	6
2	div r4, r4, r2	2	7	15
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

RS von
Mul/Div-Unit voll

Füllstand der Reservation Stations:

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	6
2	div r4, r4, r2	2	7	15
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

Erster
Add/Sub-Befehl

Füllstand der Reservation Stations:

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	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			
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

Zweiter
Add/Sub-Befehl
Hängt auch von
R4 ab $\Rightarrow$ kann
nicht vorgezogen
werden

Füllstand der Reservation Stations:

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	15
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

Stall vor Takt 7:
In-Order Issue
⇒ Es geht erst weiter, wenn Befehl 5 zugeteilt werden kann

Füllstand der Reservation Stations:

Unit	Takt													
	0	1	2	3	4	7								
Add/Sub	0	0	0	1	2	2								
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	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			
8	mul r1, r2, r3			
9	add r3, r3, r3			
10	sub r4, r4, r1			

Cycle
8

Zuteilung von
Befehl 6 - hängt
von Befehl 4 ab

Füllstand der Reservation Stations:

Unit	Takt													
	0	1	2	3	4	7	8							
Add/Sub	0	0	0	1	2	2	3							
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			
9	add r3, r3, r3			
10	sub r4, r4, r1			

Cycle
9

Zuteilung von
Befehl 7 - hängt
von Befehl 5 ab
RS voll $\Rightarrow$ Befehl
3 wird als erstes
ausgeführt

Füllstand der Reservation Stations:

Unit	Takt													
	0	1	2	3	4	7	8	9						
Add/Sub	0	0	0	1	2	2	3	4						
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		
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

Befehl 2 fertig
 ⇒ Dispatch von
 Befehl 3
 Zuteilung von
 Befehl 8

Füllstand der Reservation Stations:

Unit	Takt															
	0	1	2	3	4	7	8	9	16							
Add/Sub	0	0	0	1	2	2	3	4	4							
Mul/Div	0	1	2	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		
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

Befehl 3 fertig
⇒ Dispatch von
Befehl 4
Zuteilung von
Befehl 9

Füllstand der Reservation Stations:

Unit	Takt															
	0	1	2	3	4	7	8	9	16	19						
Add/Sub	0	0	0	1	2	2	3	4	4	4						
Mul/Div	0	1	2	2	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
22

Befehl 4 fertig
⇒ Dispatch von
Befehl 5 und 6
Zuteilung von
Befehl 10

Füllstand der Reservation Stations:

Unit	Takt														
	0	1	2	3	4	7	8	9	16	19	22				
Add/Sub	0	0	0	1	2	2	3	4	4	4	4				
Mul/Div	0	1	2	2	2	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	33
8	mul r1, r2, r3	16		
9	add r3, r3, r3	19		
10	sub r4, r4, r1	22		

Cycle
31

Befehl 5 und 6
fertig
⇒ Dispatch von
Befehl 7

Füllstand der Reservation Stations:

Unit	Takt												
	0	1	2	3	4	7	8	9	16	19	22	31	
Add/Sub	0	0	0	1	2	2	3	4	4	4	4	3	
Mul/Div	0	1	2	2	2	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	38
9	add r3, r3, r3	19	34	36
10	sub r4, r4, r1	22		

Cycle
34

Befehl 7 fertig
 ⇒ Dispatch von
 Befehl 8 und 9

Füllstand der Reservation Stations:

Unit	Takt													
	0	1	2	3	4	7	8	9	16	19	22	31	34	
Add/Sub	0	0	0	1	2	2	3	4	4	4	4	3	1	
Mul/Div	0	1	2	2	2	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	38
9	add r3, r3, r3	19	34	36
10	sub r4, r4, r1	22	39	41

Cycle
39

Befehl 8 und 9
fertig
⇒ Dispatch von
Befehl 10

Füllstand der Reservation Stations:

Unit	Takt														
	0	1	2	3	4	7	8	9	16	19	22	31	34	39	41
Add/Sub	0	0	0	1	2	2	3	4	4	4	4	3	1	1	0
Mul/Div	0	1	2	2	2	2	2	2	2	2	2	1	1	0	0

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
- Volles Bypassing
- FP-Register und normale Register können gleichzeitig in der WB-Stufe beschrieben werden
- Schreiben in Speicher geschehe nicht über CDB

Annahmen & Voraussetzungen (Forts.)

- 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

Ausführungseinheiten

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

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

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 $R2 = R1 + 24$

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

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
-		

Registerstatustabelle:

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

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	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

Befehlsfenster nach Takt 1 & 2

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

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

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-Add	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-Add			L/S 1		FP-Add

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

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-Add	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-Add	L/S 1		FP-Add

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	ID	
SUB	R3,R1,R2				IF		
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

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.

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-Add	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						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				
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			
ADD	R1,R1,#8				IS	EX	WB				
SUB	R3,R1,R2						IS	EX	WB		
BLTZ	R3,LOOP								IS	EX	WB

Leistung

- 18 Befehle in 22 Takten
- Durchsatz = $\frac{18}{22}$, IPC $\approx 0,82$

SimpleScalar – sim-outorder – Aufgabe 4

Der Simulator `sim-outorder` aus dem SimpleScalar Toolset ermöglicht die zyklengenaue Simulation von komplexen, superskalaren Prozessoren. Der Prozessor selbst kann über eine Vielzahl von Optionen parametrisiert werden, so ist z.B. möglich die Anzahl der Ausführungseinheiten oder die Zuordnungsbreite zu verändern.

Gegeben seien nun die Benchmarks `basicmath`, `qsort` und `susan` aus der MiBench Benchmark-Suite. Simulieren Sie nun diese drei Benchmarks mit Hilfe des `sim-outorder` Simulators. Variieren Sie die Anzahl der Integer- und Floating Point-ALU-Einheiten zwischen 1 und 3. Die Anzahl der Integer-ALUs können über den Parameter `-res:ialu`, die Anzahl der FP-ALUs über den Parameter `-res:fpalu` verändert werden.

Was können Sie bei diesen Benchmarks beobachten?

Config		Benchmark (in 10^6 Zyklen)		
Int	FP	basicmath	qsort	susan -s
1	1	235,3	52,8	22,8
1	2	235,3	52,8	22,8
1	3	235,3	52,8	22,8
2	1	192,7	35,2	15,6
2	2	192,7	35,2	15,6
2	3	192,7	35,2	15,6
3	1	185,6	32,3	14,9
3	2	185,6	32,2	14,9
3	3	185,6	32,2	14,9

Config		Benchmark (in 10^6 Zyklen)		
Int	FP	basicmath	qsort	susan -s
1	1	235,3	52,8	22,8
1	2	235,3	52,8	22,8
1	3	235,3	52,8	22,8
2	1	192,7	35,2	15,6
2	2	192,7	35,2	15,6
2	3	192,7	35,2	15,6
3	1	185,6	32,3	14,9
3	2	185,6	32,2	14,9
3	3	185,6	32,2	14,9

#Zyklen skaliert mit Anzahl der Int-Ausführungseinheiten

#FPs hat fast keinen Einfluss

→ 3 Int + 1 FP zu wählender Kompromiss bezgl.
HW-Aufwand und erhaltener Leistung

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
----------	----------	----------	----------	------------

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.

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

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 5

```
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 5

```
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 5

```
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 5

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 5

```
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 5

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 5

```
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 5

```
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		

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 5

```
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 5

```
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		

```
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 5b)

```
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 5b)

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 5b)

```
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 5b)

```
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 5b)

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 5b)

```
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 5b)

```
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 5b)

```
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 5b)

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 5b)

```
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 5b)

```
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 5b)

```
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

Fragen?

Fragen?