Pipelining (2)

Ergebnis der Zwischen-Evaluation (1/4)

	Schnitt		1	2	3	4	5	N/W
1. Die Vorlesung ist inhaltlich interessant	2,3		11%	58%	26%	3%	3%	0%
2. Ich besuche die Vorlesung gerne	2,4		8%	61%	21%	8%	3%	0%
3. Der Dozent hält die Vorlesung selbst gerne	1,6		47%	37%	11%	0%	0%	5%
4. Der Dozent kennt sich mit dem Vorlesungsthema aus	1,4		63%	32%	3%	3%	0%	0%
5. Der Dozent drückt sich klar und verständlich aus	1,6		42%	53%	5%	0%	0%	0%
6. Die Vorlesung ist gut strukturiert	1,9		29%	53%	8%	8%	0%	3%
7. Der Dozent vermittelt die Schlüssel-konzepte, Begriffe und Methoden	1,9		34%	47%	16%	3%	0%	0%
8. Der Dozent geht während oder nach der Vorlesung auf Fragen ein	1,2		74%	21%	0%	0%	0%	5%
9. Fragen werden zufriedenstellend beantwortet	1,7		37%	53%	5%	0%	0%	5%
10. Der Dozent ist regelmäßig für Fragen verfügbar	1,7		50%	18%	11%	8%	0%	13%

Ergebnis der Zwischen-Evaluation (2/4)

	Schnitt		1	2	3	4	5	N/W
11. Die Übungen prüfen mein Verständnis und Wissen vernünftig	2,2		18%	47%	24%	3%	3%	5%
12. Die Übungen können im verfügbaren Zeitrahmen von angemessen vorbereiteten Studenten bewältigt werden	1,9		32%	37%	21%	3%	0%	8%
13. In der Vorlesung lerne ich etwas Neues dazu	1,9		42%	34%	21%	0%	3%	0%
14. Ich würde diese Vorlesung anderen Studenten empfehlen	1,9		29%	53%	11%	3%	3%	3%
15. Ich würde den Dozenten anderen Studenten empfehlen	1,7		42%	42%	16%	0%	0%	0%
16. Die Vorlesung regt mich dazu an, mich auch weiter mit dem Thema zu beschäftigen	2,6		8%	42%	34%	13%	3%	0%

Ergebnis der Zwischen-Evaluation (3/4)

	Schnitt		1	2	3	4	5	N/W
17. Die Bereitstellung der Audiodateien (Vorlesungsmitschnitte) war hilfreich	1,6		42%	32%	5%	3%	0%	5%
18. Das Tempo in den Vorlesungen ist (1=viel zu langsam, 5=viel zu hoch)			0%	29%	63%	5%	0%	0%
19. Der Stoffumfang ist (1=viel zu gering, 5=viel zu groß)			0%	11%	66%	24%	0%	0%
20. Der Schwierigkeitsgrad ist (1=viel zu niedrig, 5=viel zu hoch)			0%	16%	68%	13%	0%	3%
21. Schulnote für Veranstaltung insgesamt	2,1		11%	71%	13%	5%	0%	0%
22. Schulnote für Dozenten in Bezug auf Fachkompetenz	1,5		55%	37%	8%	0%	0%	0%
23. Schulnote für Dozenten in Bezug auf Didaktik	1,9		26%	58%	13%	3%	0%	0%

(Basis: 38 ausgefüllte Fragebogen)

Ergebnis der Zwischen-Evaluation (4/4)

+	– / Verbesserungen
• Audiodateien und Folien online (9 x) • Python (4 x) • Dozent / Eingehen auf Fragen / Interaktion Studenten-Dozent (4 x) • verständliche Ausdrucksweise / gute Veranschaulichung (3 x) • Gut strukturiert / Gute Folien (3 x) • Musterlösungen in Vorlesung • Gutes Unterrichtsklima • Pause • Tiefergehende Themen, z.B. Pipelining • interessante Übungen • aktuell	• Lärm nach Pause / zu lange Pausen (3 x) • Disziplin / mehr Autorität (3 x) • Python (3 x) • Einführung in die Übungen (2 x) • Nur eine Programmiersprache (2 x) • teilweise eintöniger Vortrag / nicht nur Folien-Vortrag (2 x) • Mehr Beispiele zu FP-Berechnung • mehr Gaming! • Exkurse zu altem Stoff kürzer • In Übungen keine umfangreiche Einführung, sondern direkt loslegen • Audiodateien mit Folien verknüpfen • oft zu lang auf einer Folie • Laberfach ohne Tiefgang • schlecht vorbereiteter Dozent

Pipeline-Hemmnisse: strukturell (1)

- Speicherzugriff (von Neumann)

	Takt 1	Takt 2	Takt 3	Takt 4	Takt 5	Takt 6	Takt 7
Befehl 1	F	D	X	M	W		
Befehl 2		F	D	X	M	W	
Befehl 3			F	D	X	M	W
Befehl 4				---	---	---	F
Befehl 5					---	---	---

- komplexe Befehle (z. B. Gleitkomma)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
SETH $3,#4000	F	D	X	M	W									
FMUL $4,$4,$3		F	D	X	X	X	X	M	W					
FADD $4,$4,$2			F	D	--	--	--	X	X	X	X	M	W	
SET $3, $10				F	--	--	--	D	--	--	--	X	M	W

Pipeline-Hemmnisse: strukturell (2)

- Beispiele für Länge der Execute-Phase komplexer Befehle (bei MMIX):
  - Integer-Multiplikation: 10 Takte
  - Integer-Division: 60 Takte
  - Gleitkommabefehle: 4 Takte (i.d.R.)
- Taktrate auf Dauer des längsten Befehls anheben?

Pipeline-Hemmnisse: strukturell (3)

- Lösung für Speicherzugriff: Prefetching
  - In jeder Fetch-Phase mehrere Befehle aus dem RAM lesen und puffern („Fetch Buffer“)
  - z. B. MMIX: Befehlslänge 4 Bytes, eine Leseoperation liefert aber 8 Bytes
  - Ungenutzte Memory-Phasen (Befehle ohne Speicherzugriff) für weitere Fetch-Operationen verwenden
- getrennte Prozessor-Caches für Befehle und Daten (→ Pseudo-Harvard-Architektur)

Pipeline-Hemmnisse: Datenabhängigkeiten (1)

- Data hazards: Befehl benötigt ein Ergebnis, das noch nicht fertig berechnet wurde
- Beispiel: Tausch von zwei Werten (aus Quicksort)

	1	2	3	4	5	6	7	
`XOR l,l,r`	F	D	X	M	W			
`XOR r,l,r`		F	D	--	--	X	M	(l nicht bekannt)
`XOR l,l,r`			F	--	--	D	--	(r nicht bekannt)
`CMP tmp,l,pivot`				--	--	F	--	

Read-after-Write- (**RAW**-) Konflikt (zweiter Befehl braucht das im ersten berechnete l)

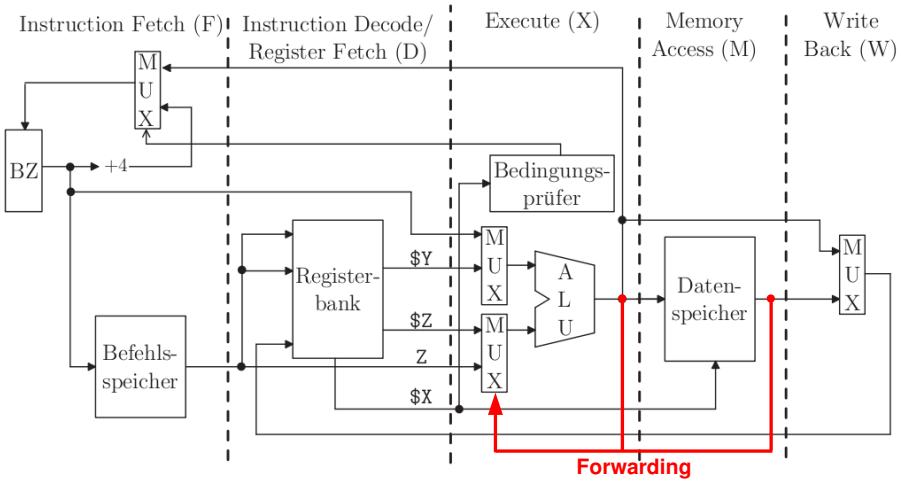
- Lösung: Result Forwarding / Bypassing

Pipeline-Hemmnisse: Datenabhängigkeiten (2)

Result Forwarding / Bypassing

- Ergebnisse von X- und M-Phasen immer als ALU-Input zurückreichen („feed back“)
- Spezielle Steuerlogik: Falls erforderlich, die zurückgereichten Ergebnisse (anstelle der Registerinhalte) verwenden

Pipeline-Hemmnisse: Datenabhängigkeiten (3)



Pipeline-Hemmnisse: Datenabhängigkeiten (4)

- ohne Forwarding

	1	2	3	4	5	6	7	
`XOR l,l,r`	F	D	X	M	W			
`XOR r,l,r`		F	D	--	--	X	M	(l nicht bekannt)
`XOR l,l,r`			F	--	--	D	--	(r nicht bekannt)
`CMP tmp,l,pivot`				--	--	F	--	

- mit Forwarding

	1	2	3	4	5	6	7
`XOR l,l,r`	F	D	X	M	W		
`XOR r,l,r`		F	D	X	M	W	
`XOR l,l,r`			F	D	X	M	W
`CMP tmp,l,pivot`				F	D	X	M

Pipeline-Hemmnisse: Datenabhängigkeiten (5)

- Result Forwarding hilft nicht immer
  - Lade-Befehle: Erst nach M-Phase steht Ergebnis zur Verfügung

```
                      1  2  3  4  5  6  7
LDO $1,base,off       F  D  X  M  W
ADD $1,$1,$2             F  D  --  X  M  W
SUB $3,$4,$5                F  --  D  X  M
```


Pipeline-Hemmnisse: ablaufbedingt (1)

- bei bedingtem Sprungbefehl: unklar, an welcher Stelle es weiter geht
- Sprungziel steht erst nach Execution-Phase des Sprungbefehls fest
- Frage: Was in die Pipeline schreiben?
  - einfach: Immer davon ausgehen, dass *nicht* gesprungen wird
  - MMIX: „probable branch“ vs. „branch“
  - komplizierter: Sprungvorhersage (eigenes Thema)

Pipeline-Hemmnisse: ablaufbedingt (2)

Beispiel: Berechne $\sum_{i=1}^{9} i$ und $2\sum_{i=1}^{9} i$

```
i       IS $1
sum     IS $2
        SET i,10
        SET sum,0
loop    SUB i,i,1
        ADD sum,sum,i
        BNZ i,loop
cont    STB sum,erg
        ADD $3,sum,sum
        STB $3,erg2
```

```
                1  2  3  4  5  6  7  8  9  10 11 12
SET i,10        F  D  X  M  W
SET sum,0          F  D  X  M  W
SUB i,i,1             F  D  X  M  W
ADD sum,sum,i            F  D  X  M  W
BNZ i,loop               F  D  X  M  W
STB sum,erg                  F  D
ADD $3,sum,sum               F
STB $3,erg2
```

Erst nach Schritt 7 ist klar, dass gesprungen wird

Zwei Stufen der Pipeline löschen

Interrupts (1)

- Auf externe (asynchrone) Interrupts muss die CPU schnell reagieren und in den Interrupt-Handler verzweigen
- Es dürfen keine Befehle unterbrochen werden, die bereits einen Teil der Zustandsänderungen bewirkt haben
- Beispiel: `STB $1, label`
  - nicht zwischen M- und W-Phase unterbrechen!
  - M-Phase: Wert aus $1 in Speicher schreiben
  - W-Phase: Falls Überlauf, Register `rA` anpassen

Interrupts (2)

- Darum Befehle in „später“ Pipeline-Phase (X oder M), noch – vor Interrupt-Behandlung – fertig bearbeiten
- Befehle, die gerade erst dekodiert werden, können aber wieder verworfen werden

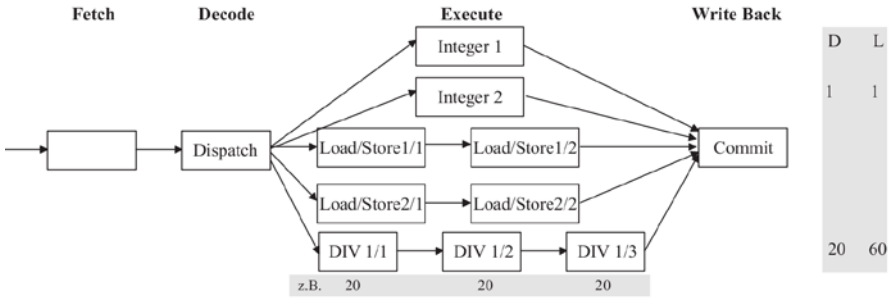
Superskalare Architekturen

Superskalare Arch. (1)

- Idee: Parallelisierung durch mehrere (potenziell) parallel arbeitende Ausführungseinheiten (Funktionseinheiten)
- Die Funktionseinheiten können für sich wiederum mehrere Pipelinestufen enthalten
- Der Prozessor „entdeckt“ Möglichkeiten zur Parallelverarbeitung (*impliziter* Parallelismus)

  (*expliziter* Parallelismus: vom Programmierer festgelegt; Intel EPIC: **E**xplicit **P**arallel **I**nstruction **C**omputing; VLIW-Befehle – **V**ery **L**arge **I**nstruction **W**ord)

Superskalare Arch. (2)



- **Durchsatzzeit** (**D**): Minimaler Abstand zwischen zwei Befehlen, die eine Ausführungseinheit verlassen
- **Latenzzeit** (**L**): Minimale Laufzeit eines Befehls durch eine Ausführungseinheit

Einführendes Beispiel: Parallele Pipelines (1/3)

- abhängige Pipelines
- unabhängige Pipelines

Beispiel: Rechnung aus Mandelbrot-Programm (siehe Anhang)

```
23          FADD   p,p,plow
24          FADD   q,q,qlow
25          SET    xk,0
26          SET    yk,0
27          SET    k,0
28      * Nächste Iteration: x_{k+1} = x_k^2-y_k^2+p
29      1H  INCL   k,1
```


Einführendes Beispiel: Parallele Pipelines (2/3)

- abhängige Pipelines (z. B. Pentium)
  - Instruktionen müssen synchron durch die Pipelines laufen: gleichzeitiger Wechsel in nächste Stufe
  - Befehle müssen also ggf. „aufeinander warten“
  - Pipelines heißen **U** und **V** (wie bei Pentium)

	1	2	3	4	5	6	7	8	9	Pipeline
FADD q,q,qlow	F	D	X	X	X	X	M	W		**U**
SET xk,0	F	D	X	**	**	**	M	W		**V**
SET yk,0		F	D	**	**	**	X	M	W	**U**
SET k,0		F	D	**	**	**	X	M	W	**V**
INCL k,1			F	**	**	**	D	X	M	**U**

Einführendes Beispiel: Parallele Pipelines (3/3)

- unabhängige Pipelines
  - nicht synchron durch die Stufen; kein Warten
  - mehr „Logistik“ für Steuerung der Stufen nötig

	1	2	3	4	5	6	7	8	Pipeline
FADD q,q,qlow	F	D	X	X	X	X	M	W	**U**
SET xk,0	F	D	X	M	W				**V**
SET yk,0		F	D	**	**	**	X	M	**U**
SET k,0		F	D	X	M	W			**V**
INCL k,1			F	**	**	**	D	X	**U**

  - ◯: „**out-of-order completion**“ (Fertigstellung in falscher Reihenfolge)

Out-of-order completion

- Problem: Ergebnis eines späten Befehls vor Fertigstellung eines vorangehenden Befehls verfügbar
- Befehle sind aber voneinander abhängig
- Pipeline muss Befehl evtl. anhalten, bis Ergebnisse aus logisch vorangehenden Befehlen vorliegen
  - kann wieder Result Forwarding verwenden

Vorschau 25.11.2010

- Superskalare Architekturen (Fortsetzung)

Anhang: Mandelbrot-Programm (1/2)

```
          PREFIX  :Mandel:
          LOC     #100
plow      GREG    #c002000000000000       -0,025
pdelta    GREG    #3f83333333333333       0,009375=(0,75- (-2,25))/320
qlow      GREG    #bff8000000000000       -1,5
qdelta    GREG    #3f8999999999999a       0,0125=(1,5 - (-1,5))/240
p         IS      $1
q         IS      $2
xk        IS      $3                      x_k
yk        IS      $4                      y_k
k         IS      $5                      Iteration k
r         IS      $6
bildx     IS      $7                      Gerätekoordinaten (integer)
bildy     IS      $8
test      IS      $9
temp1     IS      $10                     Zwischenergebnisse
temp2     IS      $11

:Mandel   FLOT    p,bildx
          FLOT    q,bildy
          FMUL    p,p,pdelta
          FMUL    q,q,qdelta
          FADD    p,p,plow
          FADD    q,q,qlow
          SET     xk,0
          SET     yk,0
          SET     k,0
* Nächste Iteration: x_{k+1}=x_k^2-y_k^2+p
1H        INCL    k,1
          FMUL    temp1,xk,xk             x_k^2
          FMUL    temp2,yk,yk             y_k^2
          FSUB    temp1,temp1,temp2
          FADD    temp1,temp1,p           x_{k+1}
```

Quelle: Böttcher, „Rechneraufbau und Rechnerarchitektur“

Anhang: Mandelbrot-Programm (2/2)

```
* y_{k+1}=2x_ky_k+q
          FMUL    yk,xk,yk
          SETH    xk,#4000                2,0 (Gleitkommawert!)
          FMUL    yk,yk,xk                2 * y_k
          FADD    yk,yk,q
          SET     xk,temp1                xk kann überschrieben werden
* r=x_{k+1}^2+y_{k+1}^2
          FMUL    temp1,xk,xk
          FMUL    temp2,yk,yk
          FADD    r,temp1,temp2
          FCMP    test,r,:M
          BNP     test,2F
          LDA     temp1,:Bmp:data         Punkt in Farbe k setzen
          SET     temp2,:WIDTH
          MUL     temp2,bildy,temp2       so viele Bytes bis y
          ADD     temp2,temp2,bildx
          STBU    k,temp1,temp2
          JMP     4F

2H        CMP     test,k,:K
          BNZ     test,1B                 noch eine Iteration
* sonst: farbe schwarz, also nichts zu tun
4H        INCL    bildx,1                 Loop zuende
          SET     test,:WIDTH
          CMP     test,bildx,test         Zeilenende?
          BNZ     test,:Mandel            nein
          SET     bildx,0                 ja: in nächste Zeile
          INCL    bildy,1
          CMP     test,bildy,:HEIGHT
          PBNZ    test,:Mandel
          POP     0,0                     Fertig!
```