Zentralübung Rechnerstrukturen im SS2008

Prozessorarchitektur

Fabian Nowak

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Institut für Technische Informatik
Lehrstuhl für Rechnerarchitektur

11. Juni 2008

Fragen?

Achtung:

Aktualisierte Folien und Musterlösung
zum 3. Aufgabenblatt, Aufgabe 1.1!

Inhalt

- Pipelining
- Sprungvorhersage
- Spekulative Befehlsausführung
- Superskalartechnik
- Optimierung

Pipelining – Motivation

Phasen bei der Befehlsausführung in einem Prozessor ausreichend disjunkt

$\Rightarrow$ Entkopplung

$\Rightarrow$ Überlappte Ausführung der Phasen möglich

$\Rightarrow$ Höherer Durchsatz, IPC-Wert, niedrigerer CPI-Wert

Pipelining – Leistungssteigerung

- Steigerung der Taktrate durch Verfeinerung der Stufen möglich (20 und mehr im Pentium 4)
- Flushing dafür dann sehr teuer
- Langsamste Stufe bestimmt maximale Geschwindigkeit
- Loop Unrolling
- Maximal ein Befehl pro Takt, also IPC=1 obere Schranke, bzw. CPI=1 untere Schranke

Pipelining – Probleme & Lösungen I

Probleme

- Abhängigkeiten im Befehlscode und Konflikte, die Hazards auslösen können
- 3 Konflikttypen: Datenabhängigkeiten, Ressourcenkonflikte, Steuerflußkonflikt
- Dadurch theroetisch maximale Beschleunigung nicht erreichbar

Pipelining – Probleme & Lösungen II

Lösungen

- **Datenabhängigkeit**: Stalling, NOPs, Befehlsumordnung, dynamische Konfliktbehandlung (Result Forwarding, Registerumbenennung, Tomasulo und Scoreboarding)
- **Ressourcenkonflikt**: Stalling, Pipelining der Ressource selbst zur Durchsatzerhöhung, Resourcenreplizierung, Übertaktung, Befehlsumordnung
- **Steuerflußkonflikte**: Stalling, NOPs, Befehlsumordnung bei verzögerten Sprüngen, Flushing, Sprungvorhersage, Prädikation

Sprungvorhersage – Motivation

Auflösung von Steuerflußkonflikten

Ursache: Nichtsequentielle Veränderung des Programmzählers durch

- Bedingte Sprünge
- Unbedingte Sprünge
- Externe, asynchrone Ereignisse wie Interrupts

Sprungvorhersage – Sprünge I

Unbedingte Sprünge

- `JMP/BRA, JSR/BSR/CALL`
- Verwendet bei Funktionsaufrufen
- Häufig als verzögerter Sprungbefehl behandelt (*delayed branch*)
  Nachfolgend wird die Pipeline mit `NOP`s oder normalen Befehlen gefüllt, die dann nicht rückgängig gemacht werden können
- Compiler-basierte Unterstüzung: Vorziehen des Sprungbefehls

Beispiel

```
LOAD r1,#5                    LOAD r1,#5
LOAD r2,#8          ⇒         JUMP somewhere
ADDC r3,r1,r2                 LOAD r2,#8
JUMP somewhere                ADDC r3,r1,r2
```

Sprungvorhersage – Sprünge II

Elimination unbedingter Sprünge

- Einfachste Lösung: **Sprünge vermeiden!**
- Ausrollen von Schleifen
  Nachteil: Codegröße, statisches Verfahren
- Nächstes Problem: Schleifengetragene Abhängigkeiten
- Einfache Form: Rekursive Abhängigkeit (**recurrence**)
- **Dependency Distance** definiert ausnutzbaren Parallelismus (wichtig für parallele Architekturen)
- **GCD-Test**: Sagt aus, daß Abhängigkeit entweder definitiv nicht, oder potentiell vorhanden ist

Sprungvorhersage – Sprünge III

Bedingte Sprünge

- `BLT/BGT/BNZ/BNGE/...`
- Ergebnis erst nach Berechung in EX-Phase
- Oben beschriebene Techniken zur Konfliktlösung
- Problem: Durchsatz in der Pipeline sinkt
- Lösung: **Sprungvorhersage**
  - statisch
  - dynamisch
- Ziel: Steuerkonflikt umgehen
- Durch Vermeidung von Pipeline Flushes
- Dadurch Steigerung des Durchsatzes in der Pipeline

Sprungvorhersage – Sprünge IV

Interrupts

- Einsprung in privilegierten Modus (*Kernel Mode*)
- Aufruf der passenden Behandlungsroutine
- Nicht vorhersagbar

Dynamische Sprungvorhersage – Prädiktoren

Ziel: Anpassung der Sprungvorhersage an laufendes Programm

Mittel: Berücksichtigung der Sprunghistorie

Übersicht

- 1-Bit-Prädiktor
- 2-Bit-Prädiktor (Sättigungszähler und Hysteresezähler)
- n-Bit-Prädiktor
- Korrelationsprädiktor
- Zweistufig adaptive Prädiktoren
- gshare, gselect
- Hybridprädiktoren

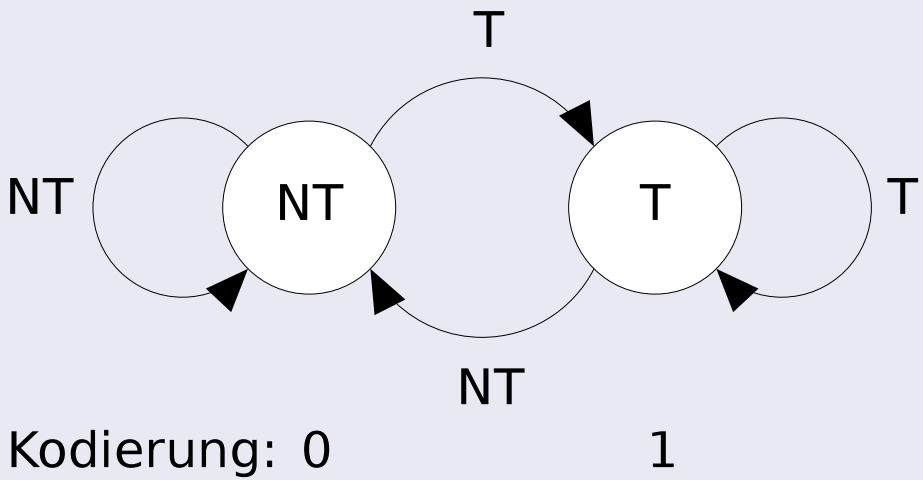
Übertreffen Vorhersagegenauigkeit statischer Prädiktoren bei weitem

Dafür: höherer **Hardwareaufwand**

Dynamische Sprungvorhersage – Vergleich I

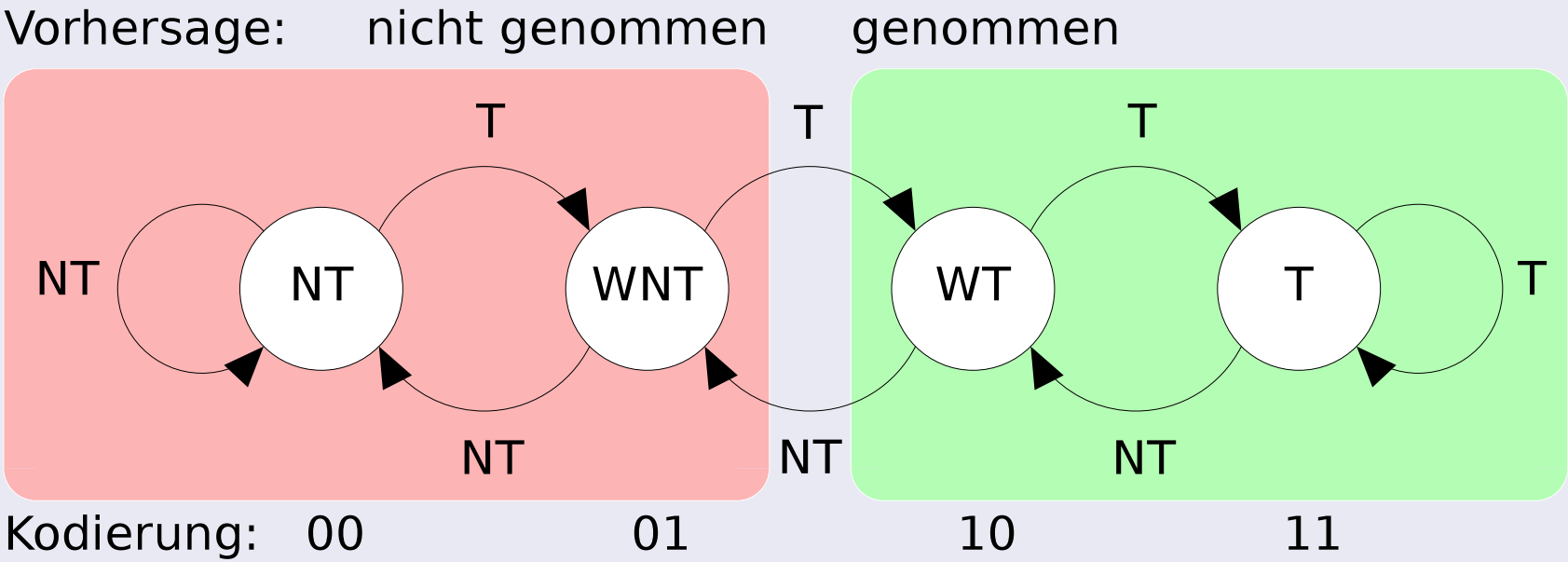
1-Bit-Prädiktor

- 2 Zustände
- Vorderstes Bit gibt Vorhersage an: 0 für *not taken* (NT), 1 für *taken* (T)
- Initialisierung maßgeblich für Vorhersage
- **Problem**: Nur kurze Historie $\Rightarrow$ Zwei Fehlvorhersagen bei verschachtelten Schleifen bei Ende und Wiedereintritt



Dynamische Sprungvorhersage – Vergleich II

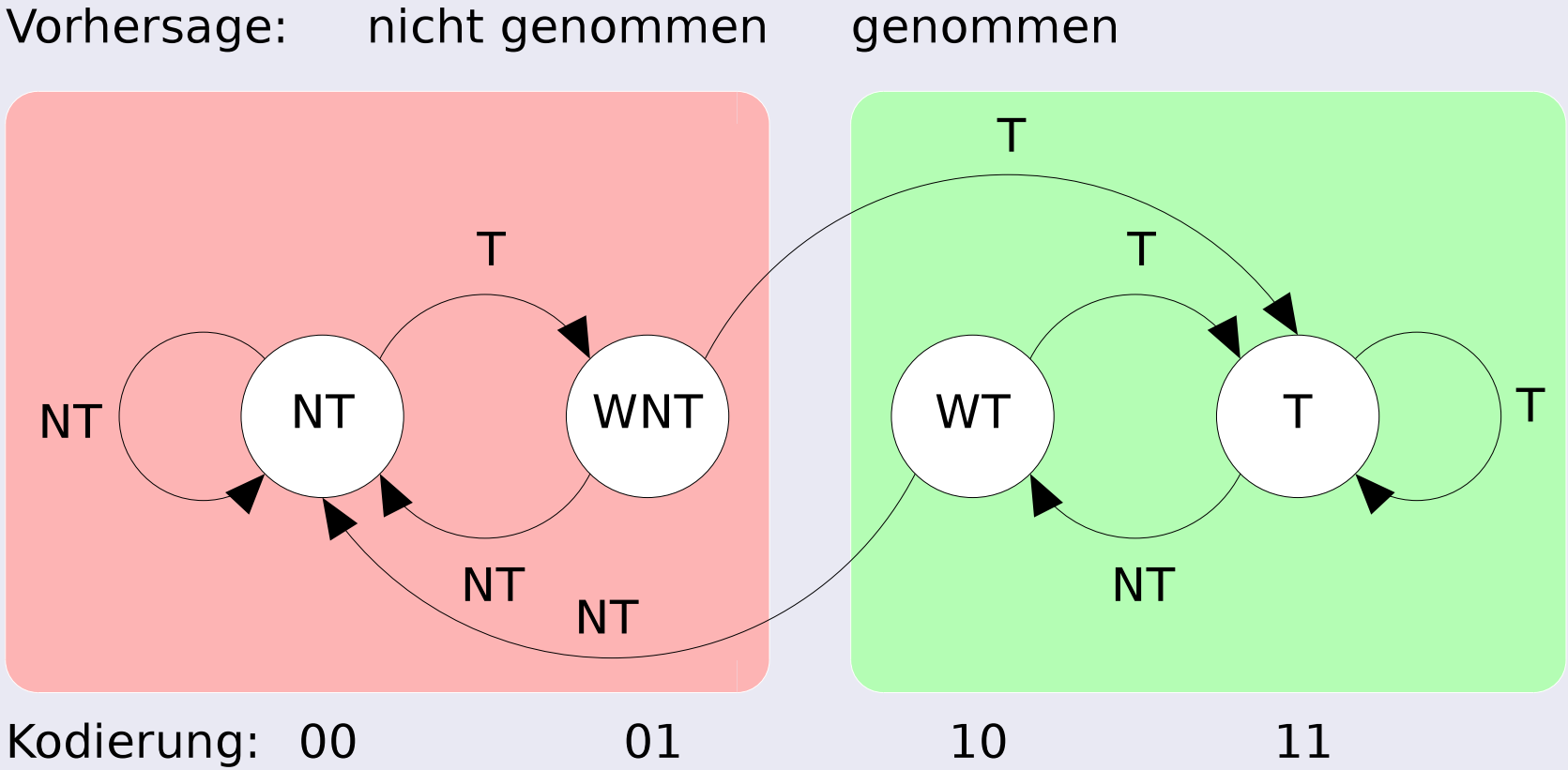
2-Bit-Prädiktor mit Sättigungszähler



- Allgemein: $2^{\text{Anzahl Bits}}$ Zustände
- Bei 2 Bits: Zusätzlich *weakly (not) taken* (W(N)T)
- Anpassung der Vorhersage durch Auf-/Abwärtszählen bei Sättigungszähler nach tatsächlichem Sprungverlauf

Dynamische Sprungvorhersage – Vergleich III

2-Bit-Prädiktor mit Hysteresezähler



- Wenige aufeinanderfolgende falsche Vorhersagen führen zu sicherem Zustand der anderen Vorhersagerichtung

Dynamische Sprungvorhersage – Vergleich IV

Korrelationsprädiktoren

- Einführung von Korrelationsprädiktoren Anfang der 90er Jahre
- Pan et al., 1992: **(m,n)-Korrelationsprädiktoren**
  (correlation-based predictors, correlating predictors)
- Yeh und Patt, 1993: **Zweistufig adaptive Pradiktoren**
  (bi-level adaptive predictors)
- McFarling, 1994: **gselect** und **gshare**
- McFarling, 1993: **Hybridprädiktoren**

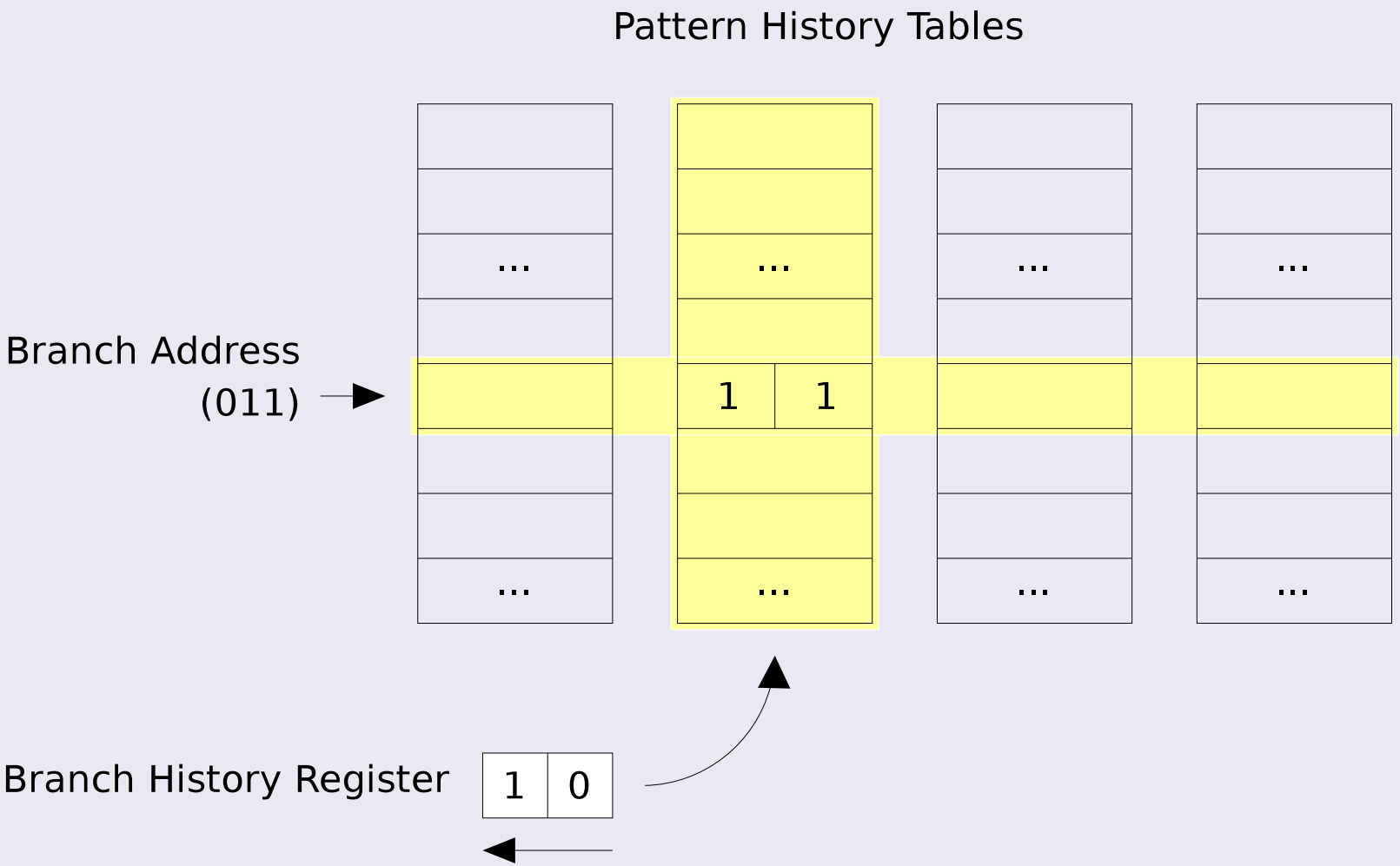
Dynamische Sprungvorhersage – Vergleich V

(m,n)-Korrelationsprädiktoren

- Einführung einer m Sprünge umfassenden Historie
- Speicherung in Sprungverlaufsregister bzw. Branch History Register (BHR)
- BHR ist m Bit breites Schieberegister, hält den Ausgang der letzten m Sprünge
- Auswahl eines n-Bit-Prädiktors aus Sprungverlaufstabelle (Pattern History Table, PHT) anhand von Sprungadresse und BHR
- (Teil der) Sprungadresse selektiert Reihe
- BHR selektiert eine von 2^{m} Spalten
- Ausgewähltes Speicherfeld enthält n-Bit-Prädiktor

Dynamische Sprungvorhersage – Vergleich VI

Beispiel: (2,2)-Korrelationsprädiktor



Dynamische Sprungvorhersage – Vergleich VII

1-/2-Bit-Prädiktor vs. Korrelationsprädiktor

- Berücksichtigung von Abhängigkeiten zwischen Sprüngen
- Korrelationsprädiktoren mehrstufig

Dynamische Sprungvorhersage – Vergleich VIII

gselect und gshare

- Wunsch: Verringerung der Interferenzen bei **Korrelationsprädiktoren**
- Verwendung von BHR und PHT wie bekannt
- **gselect**: Adressierung durch Konkatenation von BHR und Sprungbefehlsadresse (bzw. Teilen hiervon)
- **gshare**
  - Statt Konkatenation bitweise Verknüpfung von BHR und Sprungbefehlsadresse mit XOR
  - Resultat: weniger Interferenzen bei gleicher Länge

Adressteil	BHR	gselect 4/4	gshare 8/8
00000000	00000001	00000001	00000001
00000000	00000000	00000000	00000000
11111111	00000000	11110000	11111111
11111111	10000000	11110000	01111111

Dynamische Sprungvorhersage – Vergleich IX

Hybridprädiktoren

- Kombination mehrerer separater Prädiktoren
- Prädiktoren haben **unterschiedliche Stärken**, d.h. sind auf jeweils andere **Klasse von Sprungbefehlen** zugeschnitten
- **Kombinations- bzw. Hybridprädiktor**
  - Besteht aus zwei unterschiedlichen Prädiktoren und Selektor-Prädiktor
  - Selektor-Prädiktor wählt für Vorhersage zu verwendenden Prädiktor aus
- **McFarling**
  - Zwei-Bit-Prädiktor mit gshare
  - PAp-Prädiktor („local predictor“) mit gshare
  - Kombination besser als der beste von beiden bzw. Einzelprädiktoren gleicher HW-Kosten
  - Beispiel Alpha 21264: SAg(10,1024) als Teil eines Hybridprädiktors

Dynamische Sprungvorhersage – Vergleich X

Hybridprädiktoren

- **Young & Smith**: Compilerbasierte statische Sprungvorhersage mit zweistufig adaptivem Prädiktor: Vermeidet Fehlvorhersage in Anlaufphase des kompletten Prädiktors
- **Grunwald et al.**: Selektorprädiktor nicht eigenständiger Prädiktor, sondern Wahrscheinlichkeit der Fehlspekulation

Sprungvorhersage beim Pipelining

Wo wird die Sprungvorhersage ausgeführt?

- Die **Befehlsholeinheit** holt in Abhängigkeit der Vorhersage die nächsten Befehle
- Die zugehörige Adresse steht im Sprungzieladress-Cache (BTAC)
- Die Berechnung der endgültigen Zieladresse erfolgt in der ALU der EX-Stufe oder einer eigenen Adressberechnungseinheit in der IF-Stufe
- Der Befehlszähler wird in der WB-Stufe geschrieben wie ein normales Register

Sprungvorhersage: Beispiel I

Code				Durchlauf 1	2	3	4
1	INIT:	LOAD R1, #0	; R1=0				
2		LOAD R2, #2	; R2=2				
3	START:	CMP R1, R0	; R1==0?				
4		BRNZ L1	; if (R1!=R0) goto L1	NT	T	NT	T
5		LOAD R1, #1	; R1=1				
6	L1:	CMP R1, R2	; R1==R2?				
7		BRNZ L2	; if (R1!=R2) goto L2	T	NT	T	NT
8		LOAD R1, #0	; R1=0				
9		BRA start	; goto START		T		T
10	L2:	LOAD R1, #2	; R1=2				
11		BRA start	; goto START	T		T	

Sprungvorhersage: Beispiel II

2-Bit-Prädiktor mit Sättigungszähler

Achtung: Hier stand die ganze Zeit über fälschlicherweise Hysteresezähler! Es werden **5** Fehlannahmen gemacht:

Sprung 1			Sprung 2		
Prädiktion	Sprung	Neue Vorh.	Prädiktion	Sprung	Neue Vorh.
WNT	NT	NT	WNT	**T**	T
NT	**T**	WNT	T	**NT**	WT
WNT	NT	NT	WT	T	T
NT	**T**	WNT	T	**NT**	WT

Sprungvorhersage: Beispiel III

(2,2)-Korrelationsprädiktor

Annahme: Über das BHR wird nur einer von vier verschiedenen 2-Bit-Prädiktoren ausgewählt.
Es werden **3** Fehlannahmen gemacht:

Sprung 1		
Prädiktion	Sprung	Neue Vorh.
(NT,NT)/WNT	NT	(NT,NT)/NT
(NT,T)/WNT	**T**	(NT,T)/WT & (T,T)/WNT
(T,NT)/WNT	NT	(T,NT)/NT & (NT,NT)/WNT
(NT,T)/WT	T	(NT,T)/T & (T,T)/NT

Sprung 2		
Prädiktion	Sprung	Neue Vorh.
(NT,NT)/NT	**T**	(NT,NT)/WNT & (NT,T)/WNT
(T,T)/WNT	NT	(T,T)/NT & (T,NT)/WNT
(NT,NT)/WNT	**T**	(NT,NT)/WT & (NT,T)/WT
(T,T)/NT	NT	(T,T)/NT & (T,NT)/NT

Spekulative Befehlsausführung I

Trace Caches

- **Feste Anzahl ausgeführter Befehle** wird nach jedem Sprung gespeichert
- Aus diesem Cache rührt nach dem nächsten Sprung der Befehlsstrom
- Sinnvoll bei **mehrfachen Verschachtelungen** und Schleifen
- Spart eventuell Speicherbandbreite: Ein Befehlsstrom aus Trache Cache, einer aus Instruction Cache
- Kann mit **mehreren Befehlsholeinheiten gleichzeitig** gut eingesetzt werden
- Bei Fehlvorhersage: **Invalidierung** der spekulativ geladenen Befehle

Spekulative Befehlsausführung II

Prädikation

- Motivation: Ausführung eines Befehls nach einem bedingten Sprung **vor Kenntnis des Sprungausgangs**
- Statusbits in der Pipeline: Zeigen Spekulation an und verhindern frühzeitiges Commit
- Ermöglicht gleichzeitige Ausführung beider Pfade eines Sprungs
- Sinnvoll bei kurzen Verzweigungen
- Sinnvoll, wo Sprung schwer vorhersagbar

Spekulative Befehlsausführung III

Mehrpfadausführung

- Gleichzeitige Ausführung **beider Pfade** eines Sprungs
- Bei **ähnlich wahrscheinlichen** Pfaden
- Verwendete Technik: **Registerumbenennung** zur Auflösung von Ausgabeabhängigkeiten
- Unnötig ausgeführte Befehle werden verworfen
- Korrekterweise ausgeführte Befehle werden gültig gemacht

VLIW vs. Superskalartechnik I

VLIW-Prozessoren

- Befehlswort aus mehreren einzelnen Befehlen zusammengesetzt
- Parallelität **explizit vom Compiler** angegeben
- Mehrere Dekodiereinheiten
- **Statisches** Konzept
- Spekulative Befehlsausführung häufig über zusätzliche *Predication Bits*
- „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	Steuerbits

VLIW vs. Superskalartechnik II

Superskalartechnik

- Pro Takt werden ein oder mehrere Befehle aus dem Programmspeicher geholt
- *Reservation Stations* und Registertabellen geben Aufschluß über Belegung, Nutzung und Zuweisung der Ausführungseinheiten
- **Konfliktauflösung:** Parallele Ausführbarkeit der Befehle dadurch von Hardware ermittelbar
- **Dynamisches** Konzept
- Weit verbreitet
- **Sequentielles** Erscheinungsbild

VLIW vs. Superskalartechnik III

Unterschiede

- Compilerbasiert gegenüber Hardware-Ansatz
- Statisch / dynamisch
- Externe Parallelität / interne Parallelität
- Interne Befehlsauführung in / außerhalb der Reihenfolge
- Bei VLIW keine Familienbildung und Parallelität schwer ausnutzbar
- Wenig/viel Hardwareaufwand

Precise Interrupts I

Präzise Unterbrechungsbehandlung

- Nötig bei Unterbrechungen (Interrupts, Exceptions) zur **Wahrung der sequentiellen Programmsemantik** in Out-of-order-Pipelines
- Alle Befehle vor der Unterbrechung werden **vollständig ausgeführt**
- Zusätzliche, bereits in der Pipeline vorhandenen Befehle müssen verworfen werden
- Nach der Behandlungsroutine wird das Programm normal fortgesetzt
- Sicht des Programms auf Prozessor bleibt **unverändert**

Precise Interrupts II

Zuständige Einheit

- Die **Befehlszuordnungsstufe** sorgt dafür, daß keine weiteren Befehle eingeladen werden
- Das **Befehlsfenster** muß geleert werden, so daß nur die Unterbrechungsbehandlungsroutine ausgeführt wird
- Die **Rückordnungseinheit** macht alle bereits zugewiesenen Befehle gültig
- Dadurch werden alle Umbenennungsregister frei
- Anschließend wird das Programm fortgesetzt

Ausführungseinheiten

Latenz

- Die Latenz ist die **zusätzliche** Zeit, bis ein Ergebnis zur Verfügung steht
- Gemessen vom Beginn des **nächsten** Takts an
- Entspricht also Wartezeit eines **darauffolgenden** Befehls
- $=$ Bearbeitungsdauer -1

Durchsatz

- Anzahl pro Takt beendeter Befehle
- Setzt sich aus **Bearbeitungsdauer** und **Anzahl Befehlen** in Pipeline zusammen
- $\text{Durchsatz} = \frac{\#\ \text{Befehle}}{\text{Dauer}}$
- Bei 6 Befehlen und einer Latenz von 5: $\frac{6}{5+1} = 1$

Vektoreinheiten

Konzept

- Ausführung **einer Operation auf mehreren Daten** gleichzeitig
- Single Instruction Multiple Data (SIMD)
- Größe der Datenwörter/Register 32 bis 256 Bits
- Verarbeitung von 2, 4 oder 8 Paketen auf replizierten Einheiten gleichzeitig
- Benötigen **zusätzlichen Registersatz**
- **Befehlssatzerweiterung** zum Laden/Speichern der Register und Anstoßen des Befehls
- Beispiele: MMX, SSE, Altivec, ...

Vektorregister:

Datenwort 1	Datenwort 2	Datenwort 3	Datenwort 4

0 127

Superskalare Pipelines – Phasen I

Zunächst: **Befehl holen**
Je nach Auslegung dann 4 bis 5 Phasen:

- Brinkschulte/Ungerer: 5 Phasen bestehend aus
  - **Dekodierung**: Erkennen der Befehlsklasse, der Adressierungsmodi usw.
  - **Registerumbenennung**: Belegen eines physischen Registers mit dem virtuellen
  - **Befehlszuordnung**: Zuordnung eines Befehls zu einer Einheitenfamilie und Anstoßen bei Verfügbarkeit der Operandenwerte; Eintragung in die Reservation Station
  - **Ausführung**: Ausführung der Rechenoperation oder des Speicherzugriffs
  - **Rückordnung**: Schreiben des physikalischen Registers
- **Dekodierung** und **Registerumbenennung** können dabei zusammengefaßt werden.
- Befehlszuordnung kann weiter unterteilt werden

Superskalare Pipelines – Phasen II

- Hennessy/Patterson: 4 Phasen bestehend aus
  - **Issue**: Zuweisung eines Befehls zu einer Einheitenfamilie mit mindestens einer freien Reservation Station; gleichzeitig Dekodierung und natürlich Eintragung in Reservation Station
  - **Read Operands**: Warten auf gültige Operandenwerte und Anstoßen der Operation
  - **Execute**: Rechenoperation bzw. Speicherzugriff
  - **Write Results**: Schreiben des physikalischen Registers
- Prinzipiell sind die Auslegungen also nicht sonderlich unterschiedlich.
- Wir verwenden die Definition nach Brinkschulte/Ungerer mit den zusammengefaßten ersten zwei Phasen.

Tomasulo vs. Scoreboarding I

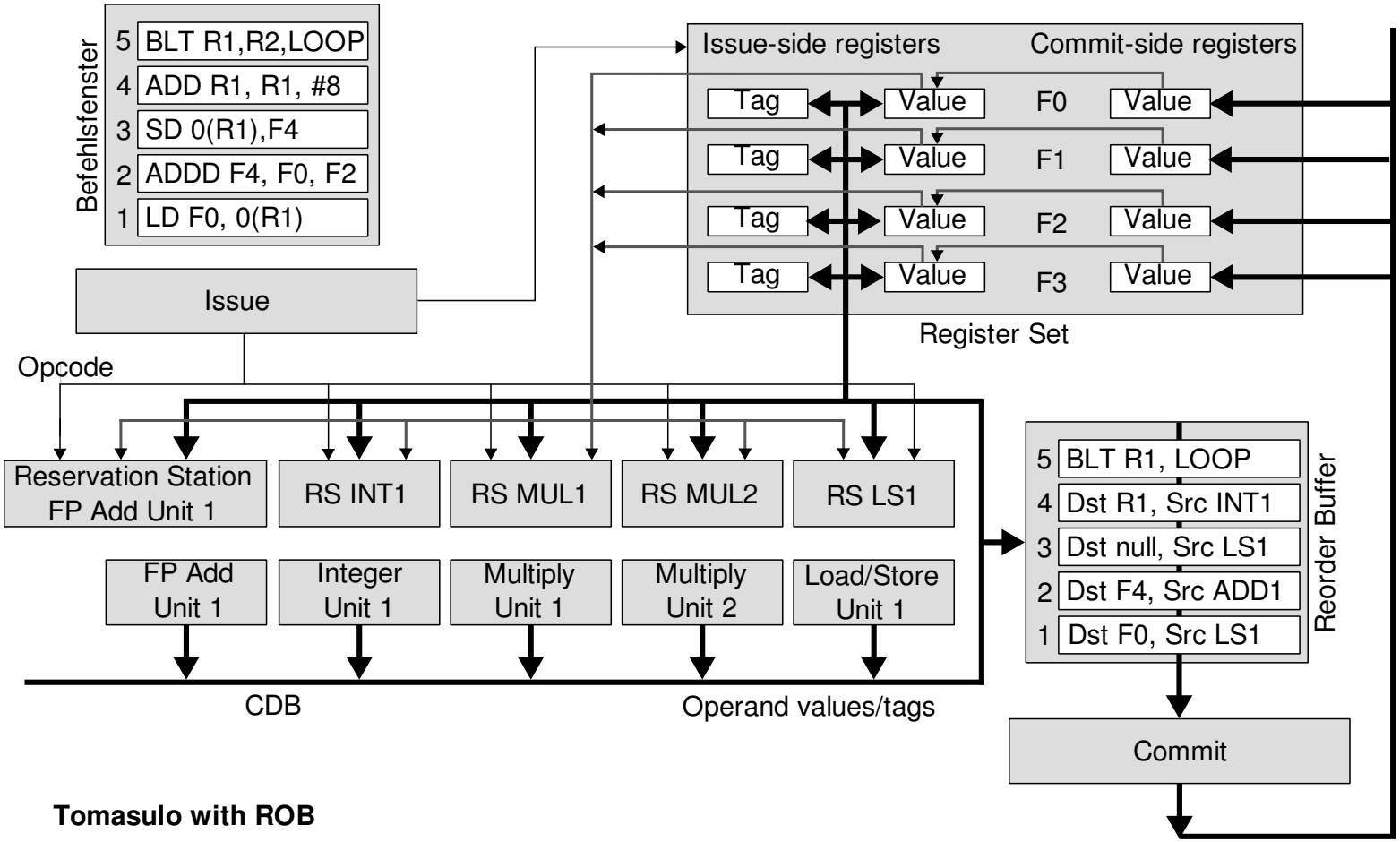
Tomasulo

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

Aktiv	Befehl	Vj	Vk	Qj	Qk	A

- Einlesen der Operanden **dezentral** in den 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)

Tomasulo vs. Scoreboarding II



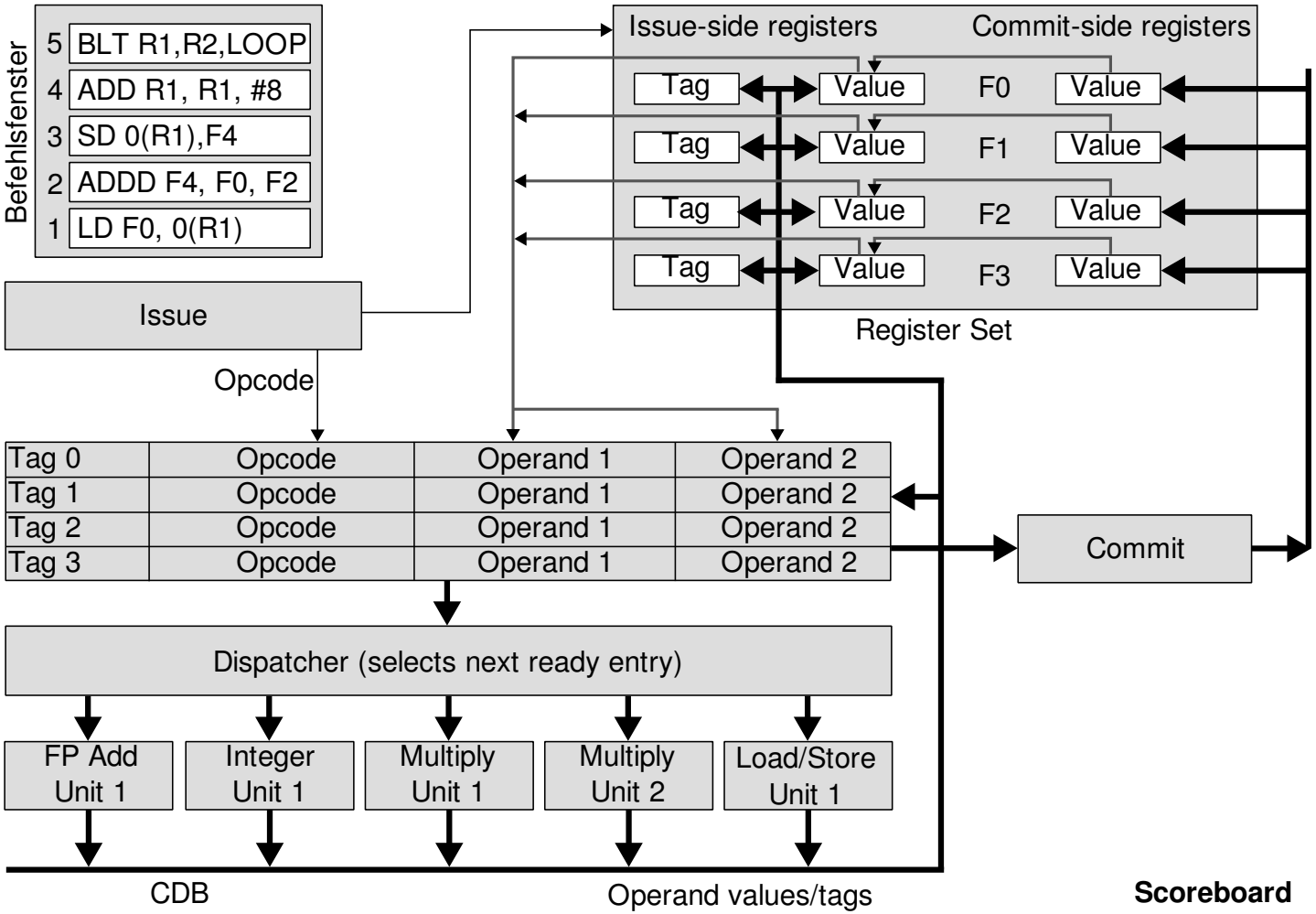
Tomasulo with ROB

Tomasulo vs. Scoreboarding III

Scoreboarding

- Bereithalten einer **begrenzten Anzahl** von Operationen zur Ausführung
- Bei Vorhandensein der Operanden erfolgt **Zuweisung an freie Einheit**
- Gemeinsamer Datenbus

Tomasulo vs. Scoreboarding IV



Tomasulo vs. Scoreboarding V

Unterschiede

- Dezentral / zentral verwaltet
- Zuweisung direkt über Reservation Station an Ausführungseinheit / getrennt
- Datenbus überlastet / entlastet

Vorteil Scoreboarding

- **Entkopplung** der Umbenennungs- und der Zuweisungseinheit von den Ausführungseinheiten
- Dadurch **Entlastung** des Datenbusses
- Dennoch: **Überwiegend Tomasulo verwendet**

Bestandteile einer Implementierung des Tomasulo-Algorithmus I

Befehlsfenster

Nummer	Befehl	Station
1	`Op  Operands`	–

- Zuständig für die Bereitstellung dekodierter Befehle an Zuweisungseinheit (**Instruction Issue**)
- Begrenzte Größe: 16 bis 46 Einträge heutzutage
- Zuweisung von bis zu sechs Anweisungen pro Takt
- Hier erweitert um Feld der derzeitigen Station; dadurch auch mehr Einträge als in Hardware

Bestandteile einer Implementierung des Tomasulo-Algorithmus II

Registerstatustabelle

Feld	PhR0	PhR1	PhR2	PhF0	PhF2	...
Architektur-Register						...
Qi						

- Gibt Auskunft über Abbildung von Architekturregistern auf physische Register
- Der Übersicht halber Valid-Bit nicht eingezeichnet, wird hier durch * angegeben

Bestandteile einer Implementierung des Tomasulo-Algorithmus III

Reservation Stations (Umordnungspuffer)

In tabellarischer Form hier zusammengefaßt:

Einheit	Aktiv	Befehl	Vj	Vk	Qj	Qk	A
L/S 1	n						
L/S 2	n						
Int-Add	n						
Int-Mul	n						
FP-Add	n						
FP-Mul	n						

Bei Tomasulo dezentral an jeder Einheit oder Familie von Einheiten.
Bei Scoreboard: Keine Unterscheidung nach Einheiten; wirklich eine zentrale Tabelle.

Bestandteile einer Implementierung des Tomasulo-Algorithmus IV

Rückordnungspuffer

Befehlsnr.	Ziel	Quelle
1	Phys. Reg.	Einheit

- Wird bei Zuweisung (Issue) eines Befehls gefüllt und enthält Zielregister sowie produzierende Einheit
- Wird von **Retirement Unit** (Rückordnungsstufe) benutzt, um **Commitment** oder **Removement** durchzuführen

$\Rightarrow$ Rückordnung der Befehle in Programmreihenfolge

Tomasulo – Beispiel I

Annahmen & Voraussetzungen

- zwei Lade-Speichereinheiten (Load/Store Unit), eine Integer-Additionseinheit, eine Integer-Multiplikationseinheit, eine FP-Additionseinheit, zwei FP-Multiplikationseinheiten und eine FP-Divisionseinheit
- Verzögerung bei bedingten Sprungbefehlen, also kein Anhalten (Stalling) und keine Vorhersage, sondern fortwährendes Füllen der Pipeline; dafür sei ein Sprungzieladresscache vorhanden und die Vorhersage laute auf Taken
- Volles Bypassing
- FP-Register und normale Register können gleichzeitig in der WB-Stufe beschrieben werden

Tomasulo – Beispiel II

Annahmen & Voraussetzungen (Forts.)

- Die Befehlszuordnungs- und Rückordnungsbandbreite 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; 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
- Speicherschreibzugriffe haben eine Ausführungsdauer von nur einem Takt

Tomasulo – Beispiel III

Annahmen & Voraussetzungen (Forts.)

- Kein Pipeling in den Ausführungseinheiten
- Dekodierte Befehle werden nur dann in die Issue-Stufe gebracht, wenn Einheit frei ist und Operanden verfügbar sind (entspricht also *Read Operands*)
- Hier zur Vereinfachung der Darstellung: Nach Dekodierung benennen wir die Stufe vor der Befehlsausführung mit Issue (IS) und zeichen sie erst, wenn die Operanden verfügbar sind. Analog für die Belegung der Reservation Stations.
- Dadurch Modellierung von Einheitenfamilien erreicht
- **Achtung:** Für den Einsatz der Registerumbenennung müssen Eintragungen in RSs eigentlich direkt nach Dekodierung erfolgen!
- Folgende Pipelinestruktur: IF ID IS EX M* (WB)

Tomasulo – Beispiel IV

Ausführungseinheiten

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

Tomasulo – Beispiel V

Auszuführender Programmcode

```
LOOP: LD     F0,0(R1)    ; loads Mem[i]
      ADDD   F4,F0,F2    ; adds to Mem[i]
      SD     0(R1),F4    ; stores into Mem[i]
      ADD    R1,R1,#8    ;
      BLT    R1,R2,LOOP  ; delayed branch if  R1 < R2
```

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

Tomasulo – Beispiel VI

Takt 1

- LD und ADDD geholt

Takt 2

- LD und ADDD dekodiert
- SD und ADD geholt
- Reservation Station noch leer

Code

```
LOOP: LD    F0,0(R1)
      ADDD  F4,F0,F2
      SD    0(R1),F4
      ADD   R1,R1,#8
      BLT   R1,R2,LOOP
```

Pipeline

Befehl		Stufe 1	Stufe 2
LD	F0,16(R1)	IF	ID
ADDD	F4,F0,F2	IF	ID
SD	0(R1),F4		IF
ADD	R1,R1,#8		IF

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
2	PhF4	FP-Add
1	PhF0	L/S 1

Registerstatustabelle:

Feld	PhR0	PhR1	PhF0	PhF2	PhF4
Architektur-Register	R1*	R2*	F0	F2*	F4
Qi					

Tomasulo – Beispiel VII

Befehlsfenster nach Takt 1 & 2

Nummer	Befehl		Station
1	`LD`	`F0,16(R1)`	ID
2	`ADDD`	`F4,F0,F2`	ID
3	`SD`	`0(R1),F4`	IF
4	`ADD`	`R1,R1,#8`	IF

Tomasulo – Beispiel VIII

Takt 3 & 4

- BLT und LD geholt, dekodiert
- ADDD und ADD geholt

Befehlsfenster:

Nr.	Befehl		Station
1	LD	F0,16(R1)	M
2	ADDD	F4,F0,F2	ID[1]
3	SD	0(R1),F4	ID[1]
4	ADD	R1,R1,#8	IS
5	BLT	R1,R2,LOOP	ID
6	LD	F0,16(R1)	ID
7	ADDD	F4,F0,F2	IF
8	SD	0(R1),F4	IF

[1]: Not yet finally issued

Code

```
LOOP: LD    F0,0(R1)
      ADDD  F4,F0,F2
      SD    0(R1),F4
      ADD   R1,R1,#8
      BLT   R1,R2,LOOP
```

Pipeline

Befehl		Stufe 1	2	3	4
LD	F0,16(R1)	IF	ID	IS	M
ADDD	F4,F0,F2	IF	ID		
SD	0(R1),F4		IF	ID	
ADD	R1,R1,#8		IF	ID	IS
BLT	R1,R2,LOOP			IF	ID
LD	F0,16(R1)			IF	ID
ADDD	F4,F0,F2				IF
SD	0(R1),F4				IF

Tomasulo – Beispiel IX

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
6	PhF6	L/S 2
5	PC	Int-Add
4	PhR2	Int-Add
3	null	any L/S
2	PhF4	FP-Add
1	PhF0	L/S 1

Reservation Stations:

Einheit	Aktiv	Befehl	Vj	Vk	Qj	Qk	A
L/S 1	j	LD					16 + [PhR0]
Int-Add	n	ADD	[PhR0]	#8			

Registerstatustabelle:

Feld	PhR0	PhR1	PhR2	PhF0	PhF2	PhF4	PhF6
Architektur-Register	R1	R2*	R1	F0	F2*	F4	F0
Qi			Int-Add	L/S 1			

Tomasulo – Beispiel X

Takt 5 & 6

- `ADD` schreibt `R1`
- Result Forwarding an `BLT` und `LD`

Befehlsfenster:

Nr.	Befehl		St.
1	`LD`	`F0,16(R1)`	M
2	`ADDD`	`F4,F0,F2`	ID
3	`SD`	`0(R1),F4`	ID
4	`ADD`	`R1,R1,#8`	WB
5	`BLT`	`R1,R2,LOOP`	IS
6	`LD`	`F0,16(R1)`	IS
7	`ADDD`	`F4,F0,F2`	ID
8	`SD`	`0(R1),F4`	ID
9	`ADD`	`R1,R1,#8`	ID
10	`BLT`	`R1,R2,LOOP`	ID
11	`LD`	`F0,16(R1)`	IF
12	`ADDD`	`F4,F0,F2`	IF

Pipeline

Befehl		Stufe 3	4	5	6
`LD`	`F0,16(R1)`	IS	M	M	M
`ADDD`	`F4,F0,F2`				
`SD`	`0(R1),F4`	ID			
`ADD`	`R1,R1,#8`	ID	IS	EX	WB
`BLT`	`R1,R2,LOOP`	IF	ID		IS
`LD`	`F0,16(R1)`	IF	ID		IS
`ADDD`	`F4,F0,F2`		IF	ID	
`SD`	`0(R1),F4`		IF	ID	
`ADD`	`R1,R1,#8`			IF	ID
`BLT`	`R1,R2,LOOP`			IF	ID
`LD`	`F0,16(R1)`				IF
`ADDD`	`F4,F0,F2`				IF

Tomasulo – Beispiel XI

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
10	PC	Int-Add
9	PhR0	Int-Add
8	null	any L/S Unit
7	PhF8	FP-Add
6	PhF6	L/S 2
5	PC	Int-Add
4	PhR2	Int-Add
3	null	any L/S Unit
2	PhF4	FP-Add
1	PhF0	L/S 1

Reservation Stations:

Einheit	Aktiv	Befehl	Vj	Vk	Qj	Qk	A
L/S 1	j	LD		16			16 + [PhR0]
L/S 2	n	LD	[PhR2]	16			
Int-Add	j	BLT	[PhR2]	[PhR1]			

Registerstatustabelle:

Feld	PhR0	PhR1	PhR2	PhF0	PhF2	PhF4	PhF6	PhF8
Architektur-Register	R1	R2*	R1*	F0	F2*	F4	F0	F4
Qi				L/S 1			L/S2	

Tomasulo – Beispiel XII

Takt 7 & 8

- Sprungvorhersage validiert
- `LD` beendet
- `ADDD` angestoßen
- Int-Einheit frei für `ADD`

Pipeline

Befehl		Stufe					
		3	4	5	6	7	8
`LD`	`F0,16(R1)`	IS	M	M	M	WB	
`ADDD`	`F4,F0,F2`					IS	EX
`SD`	`0(R1),F4`	ID					
`ADD`	`R1,R1,#8`	ID	IS	EX	WB		
`BLT`	`R1,R2,LOOP`	IF	ID		IS	EX	WB
`LD`	`F0,16(R1)`	IF	ID		IS	M	M
`ADDD`	`F4,F0,F2`		IF	ID			
`SD`	`0(R1),F4`		IF	ID			
`ADD`	`R1,R1,#8`			IF	ID		IS
`BLT`	`R1,R2,LOOP`			IF	ID		
`LD`	`F0,16(R1)`				IF	ID	
`ADDD`	`F4,F0,F2`				IF	ID	
`SD`	`0(R1),F4`					IF	ID
`ADD`	`R1,R1,#8`					IF	ID
`BLT`	`R1,R2,LOOP`						IF

Tomasulo – Beispiel XIII

Befehlsfenster:

Nr.	Befehl		St.
1	ADDD	F4,F0,F2	EX
2	SD	0(R1),F4	ID
3	BLT	R1,R2,LOOP	WB
4	LD	F0,16(R1)	M
5	ADDD	F4,F0,F2	ID
6	SD	0(R1),F4	ID
7	ADD	R1,R1,#8	EX
8	BLT	R1,R2,LOOP	ID
9	LD	F0,16(R1)	ID
10	ADDD	F4,F0,F2	ID
11	SD	0(R1),F4	ID
12	ADD	R1,R1,#8	ID
13	BLT	R1,R2,LOOP	IF

Rückordnungspuffer:

Befehlsnr.	Ziel	Quelle
12	PhR1	Int-Add
11	null	any L/S Unit
10	PhF12	FP-Add
9	PhF10	any L/S Unit
8	PC	Int-Add
7	PhR0	Int-Add
6	null	any L/S Unit
5	PhF8	FP-Add
4	PhF6	L/S 2
3	PC	Int-Add
2	null	any L/S Unit
1	PhF4	FP-Add

Tomasulo – Beispiel XIV

Reservation Stations:

Einheit	Aktiv	Befehl	Vj	Vk	Qj	Qk	A
L/S 1	n						
L/S 2	j	LD		16			16 + [PhR2]
Int-Add	n	ADD	[PhR2]	#8			
Int-Mul	n						
FP-Add	j	ADDD	[PhF0]	[PhF2]			

Registerstatustabelle:

Feld	PhR0	PhR1	PhR2	PhR3	PhF0	PhF2	PhF4
Architektur-Register	R1	R2*	R1*	R1	F0*	F2*	F4
Qi	Int-Add						FP-Add

PhF6	PhF8	PhF10	PhF12
F0	F4	F0	F4
L/S2			

Tomasulo – Beispiel XV

Befehl		Stufe											
		1	2	3	4	5	6	7	8	9	10	11	12
LD	F0,16(R1)	IF	ID	IS	M	M	M	WB					
ADDD	F4,F0,F2	IF	ID					IS	EX	EX	WB		
SD	0(R1),F4		IF	ID							IS	M/WB	
ADD	R1,R1,#8		IF	ID	IS	EX	WB						
BLT	R1,R2,LOOP			IF	ID		IS	EX	WB				
LD	F0,16(R1)			IF	ID		IS	M	M	M		WB[1]	
ADDD	F4,F0,F2				IF	ID						IS	EX
SD	0(R1),F4				IF	ID							
ADD	R1,R1,#8					IF	ID		IS	EX			WB
BLT	R1,R2,LOOP					IF	ID						IS
LD	F0,16(R1)						IF	ID					IS
ADDD	F4,F0,F2						IF	ID					
SD	0(R1),F4							IF	ID				
ADD	R1,R1,#8							IF	ID				
BLT	R1,R2,LOOP								IF	ID			

Tomasulo – Beispiel XVI

9	10	11	12	13	14	15	16	17	18	19	20
EX	WB IS	M/WB									
M		WB[1] IS	EX	EX	WB IS	M/WB					
EX			WB IS	EX	WB[2]						
			IS	M	M	M	WB IS	EX	EX	WB IS	M/WB
					IS	EX		WB IS	EX		WB[3]
ID											

Tomasulo – Beispiel XVII

Leistung

- 15 Befehle in 20 Takten
- Bei einer minimalen Pipelinelänge von 4 Stufen ergibt sich ein IPC von $\frac{15}{20-(4-1)} = \frac{15}{17} = 0.88$

Erläuterungen/Fußnoten

- [1] Der CDB wird beim Speicherzugriff nicht benutzt.
- [2] Der Befehlszähler wird nicht über den CDB geschrieben.
- [3] Hier findet tatsächlich überhaupt kein Zugriff auf den CDB statt.

Tomasulo – Beispiel XVIII

Hinweis

- Beispiel rein künstlich und nur zur prinzipiellen Darstellung
- Reservation Stations wurden **nicht** sofort nach Dekodierung gefüllt gezeichnet
- Implementierungen mit zu Familien zusammgefaßten Einheiten gleichermaßen wie ohne Zusammenfassung machbar
- Reservation Stations haben einen **oder mehrere** Einträge

Optimierung: Pipeline-gerechte Schleifen I

Vorgehen

- Erste Beobachtung: `ADD` kann verschoben werden, wenn die Adressierungen entsprechend angepaßt werden.
- Zweite Beobachtung: Behebung von echten Abhängigkeiten über Schleifengrenzen hinweg möglich
- Damit Speichern des addierten Werts, während zuletzt geladener Wert summiert wird und neuer Wert geladen wird.

Optimierung: Pipeline-gerechte Schleifen II

Schleife (R2 entsprechend verkleinert):

```
LOOP: SD    -8(R1),F4  ; buffering resolves WAR conflict
      ADD   R1,R1,#8   ;
      ADDD  F4,F0,F2   ; buffering resolves WAR conflict
      LD    F0,0(R1)   ; loads Mem[i]
      BLT   R1,R2,LOOP ; delayed branch if  R1 < R2
```

Initialisierung (Prolog):

```
      LD    F0,0(R1)   ; loads Mem[R1]
      ADD   R1,R1,#8   ;
      ADDD  F4,F0,F2   ; adds to F0
      LD    F0,0(R1)   ; loads Mem[R1]
```

Finalisierung (Epilog):

```
      SD    -8(R1),F4 ; buffering resolves WAR conflict
      ADDD  F4,F0,F2  ; adds to F0
      SD    0(R1),F4  ; buffering resolves WAR conflict
```

Optimierung: Pipeline-gerechte Schleifen III

Pipeline-Auslastung

- Floating-Point-Befehle können völlig gleichzeitig ausgeführt werden
- Einzig Warten auf `R1` nach Addition

Befehl	1	2	3	4	5	6	7	8	9	10	11	12	13
LD	IF	ID	IS	M	M	M	WB						
ADD	IF	ID	IS	EX	WB								
ADDD		IF	ID				IS	EX	EX	WB			
LD		IF	ID		IS	M	M	M	WB				
SD			IF	ID						IS	M		
ADD			IF	ID	IS	EX	[1]	WB					
ADDD				IF	ID				IS	EX	EX	WB	
LD				IF	ID			IS	M	M	M	[3]	WB
BLT					IF	ID		IS	EX	WB[2]			

[1] Strukturkonflikt auf CDB.

[2] Der PC wird nicht über den CDB geschrieben.

[3] Strukturkonflikte auf CDB und Registersatz vor.

Optimierung: Pipeline-gerechte Schleifen IV

Leistung

- 10 Befehle in 13 Takten
- Bei einer minimalen Pipelinelänge von 4 Stufen ergibt sich ein IPC von $\frac{10}{13-(4-1)} = \frac{10}{10} = 1$
- Nur die Schleife: $IPC = \frac{5}{11-(4-1)} = \frac{5}{8} = 0,625$
- Zum Vergleich: zweite Schleife der unoptimierten Version: $IPC = \frac{5}{13-(4-1)} = \frac{5}{10} = 0,5$
- Dritte Schleife der unoptimierten Version: $IPC = \frac{5}{15-(4-1)} = \frac{5}{12} = 0,417$

Fragen?

Aktualisierte Folien und Musterlösung
zum 3. Aufgabenblatt, Aufgabe 1.1!