TECHNISCHE UNIVERSITÄT DRESDEN

VLSI-EDA

Fakultät Informatik, Institut für Technische Informatik, Professur VLSI-Entwurfssysteme, Diagnostik und Architektur

Rechnerstrukturen und -organisation

Nutzung von Parallelität

Rainer G. Spallek

TU Dresden, 27.01.2022

Rainer.Spallek@tu-dresden.de http://vlsi-eda.inf.tu-dresden.de

Gliederung

1 Zielstellung

- Möglichkeiten der Nutzung von Parallelität innerhalb der CPU.
- Bewertung der Leistungsfähigkeit der CPU, Verarbeitungsleistung.
- Vergleich der einzelnen Parallelisierungsvarianten und Bewertung.
- Analyse der durch die Parallelisierung auftretenden Probleme und deren Behandlung.
- Welche Leistungssteigerungen sind theoretisch möglich, welche praktisch nutzbar.

2 Ebenen der Parallelität

Bit-Ebene (Bit Level Parallelism - BLP)

Zusammenfassung mehrerer Bits (Paralleladdierer)

Wort-Ebene (Word Level Parallelism - WLP)

Single Instruction Multiple Data - SIMD (SSE, MMX)

Befehls-Ebene (Instruction Level Parallelism - ILP)

Pipelining-, Superskalar-, VLIW-Architektur

Kontrollfluss-Ebene (Thread Level Parallelism - TLP)

Multithreaded-Architektur

Programm-Ebene (Program Level Parallelism - PLP)

Multiprocessor-, Multicore-, Multicomputer-Architektur

Bewertung der Leistungsfähigkeit von Rechnern

CPI - Anzahl der Taktzyklen pro Befehl (Cycles Per Instruction)

IPC - Abgearbeitete Befehle pro Taktzyklus (Instruction Per Cycle)

T_{EXE} - Abarbeitungszeit

T_C - Taktzykluszeit (Taktperiodendauer)

f_C - Taktfrequenz

N - Anzahl der abzuarbeitenden Befehle

Abarbeitungszeit

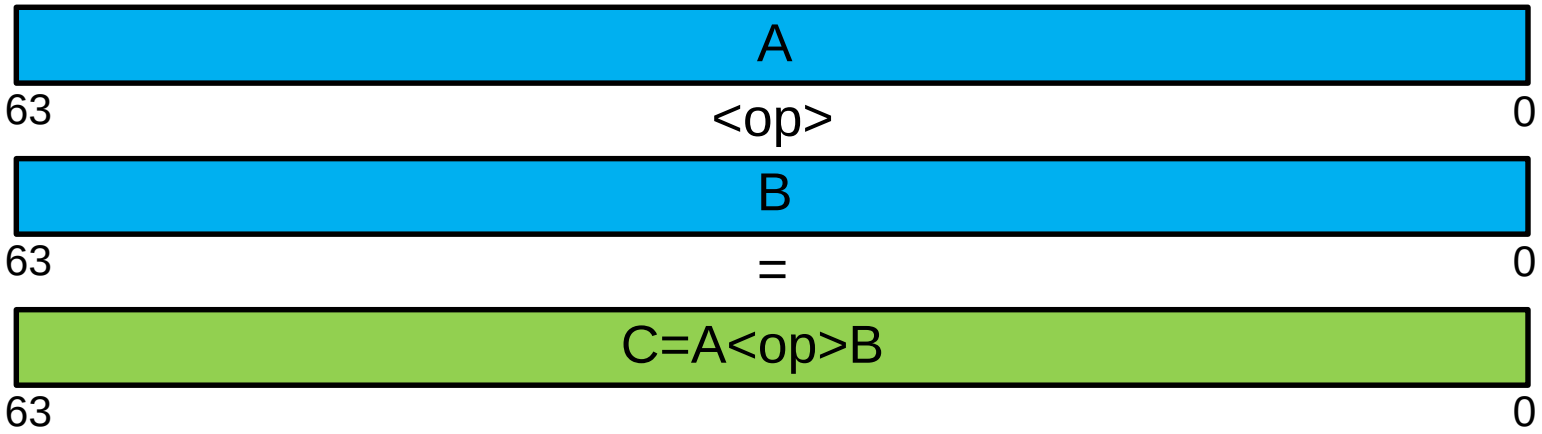
$$T_{EXE} = N \cdot CPI \cdot T_C \qquad T_{EXE} = \frac{N \cdot T_C}{IPC} \qquad f_C = \frac{1}{T_C}$$

CPI, IPC - charakteristische Durchschnittswerte – architekturspezifisch

Die Bewertung der Verarbeitungsleistung von Datenpfaden erfolgt typisch durch Cycles Per Instruction *CPI* und die mögliche Taktfrequenz f_C.

3 Parallelverarbeitung auf Bit-Ebene

64-bit Paralleloperation

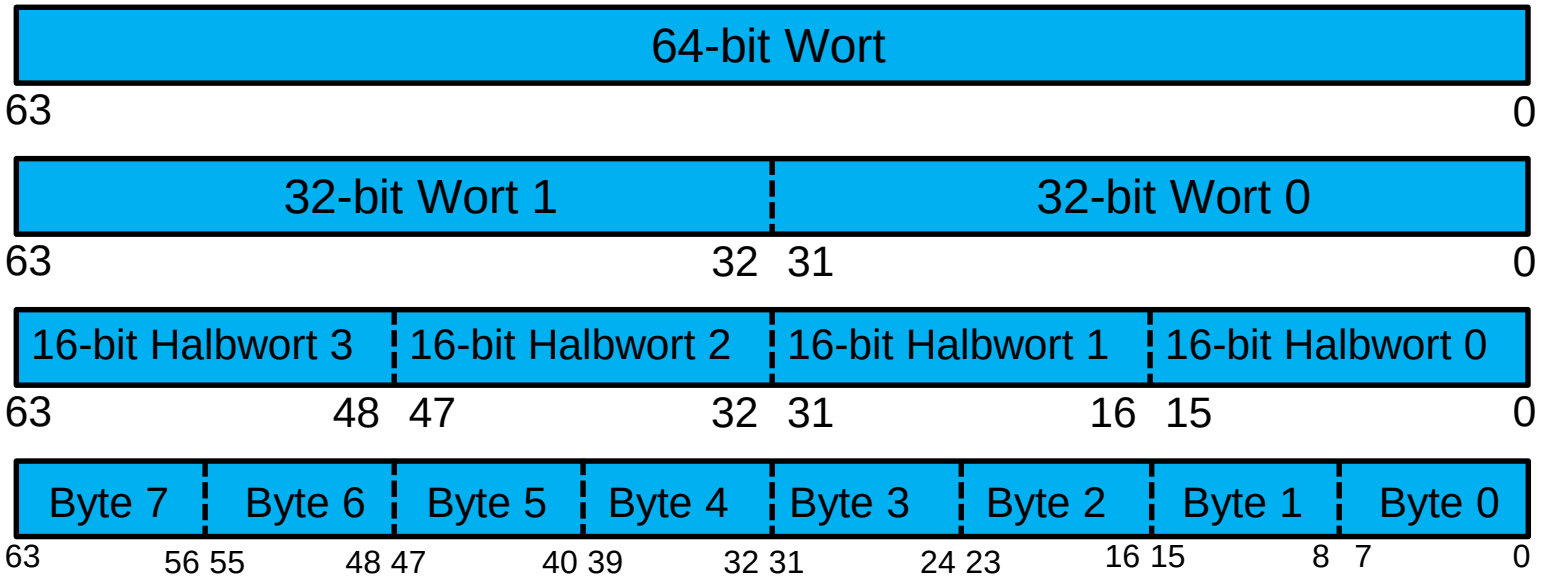


Bitparallele Verarbeitung des Darstellungsformates – in einem Taktzyklus
Parallele Addierer, Multiplizierer, Barrelshifter, Boolesche Funktionen, …
→ Ein Befehl mit 64 bit breiten Operanden pro Taktzyklus.

CPI = 1

4 Parallelverarbeitung auf Wort-Ebene

Subwortunterteilung eines 64-bit Wortes



Unterteilung eines n-bit Wortes lückenlos in n/m bit Subworte gleicher Größe.

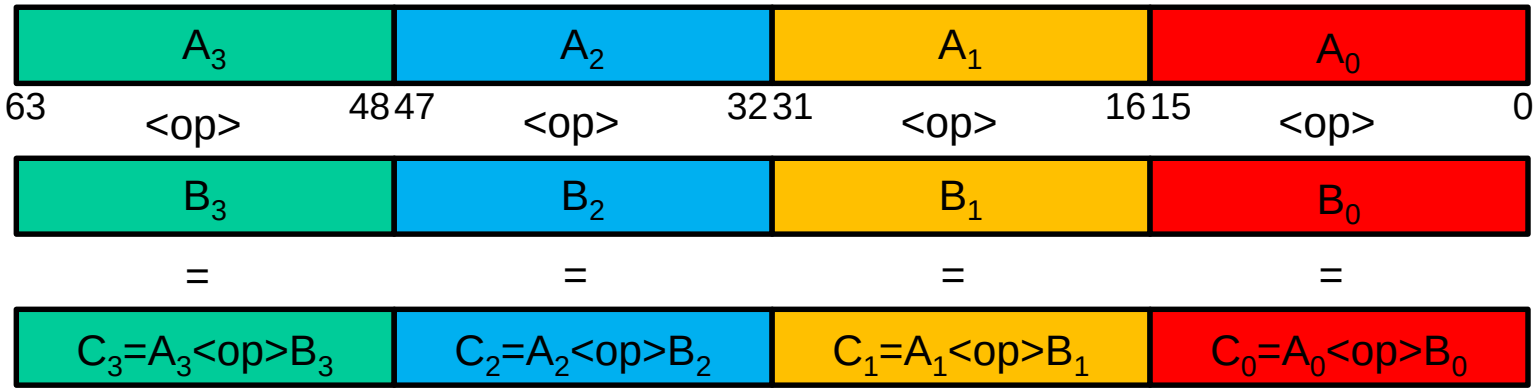
Mit $n=2^i$, $i=1,2,3, \ldots$; $m=2^j$, $j=1,2,3, \ldots$ und $i \geq j$

SIMD Prinzip (Single Instruction, Multiple Data)

Parallelverarbeitung von vier 16-bit Subworten

Anwendung eines Befehls (Operation) auf mehrere Daten (Subwörter)

MMX, SSE, 3DNow, AltiVec, SSE2, SSE3, SSSE3, SSE4, SSE5 und AVX.



Auftrennung der Übertragsweiterleitung an den Subwortgrenzen.

→ Ein Befehl für vier gleiche Operationen mit 16 bit Subworten pro Taktzyklus

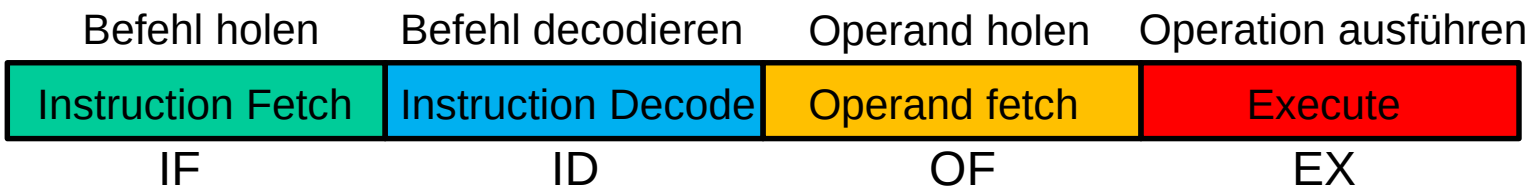
CPI = 0,25

5 Parallelverarbeitung auf Befehls-Ebene

Varianten paralleler Abarbeitung mehrere Befehle gleichzeitig

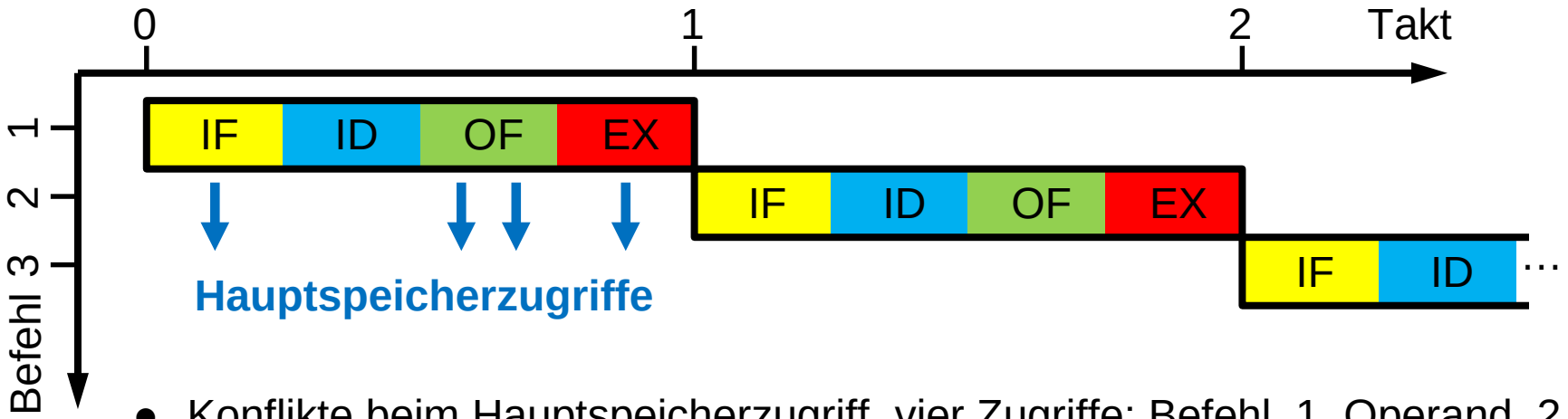
- Pipelining
- Superpipelining
- Superscalar
- VLIW (Very Long Instruction Word)
- Mehrkern-Prozessoren
- Mehrprozessor-Architekturen

Phasen des Befehlszyklus (von Neumann)



5.1 Befehls-Pipelining

Eintakt-Befehlsabarbeitung

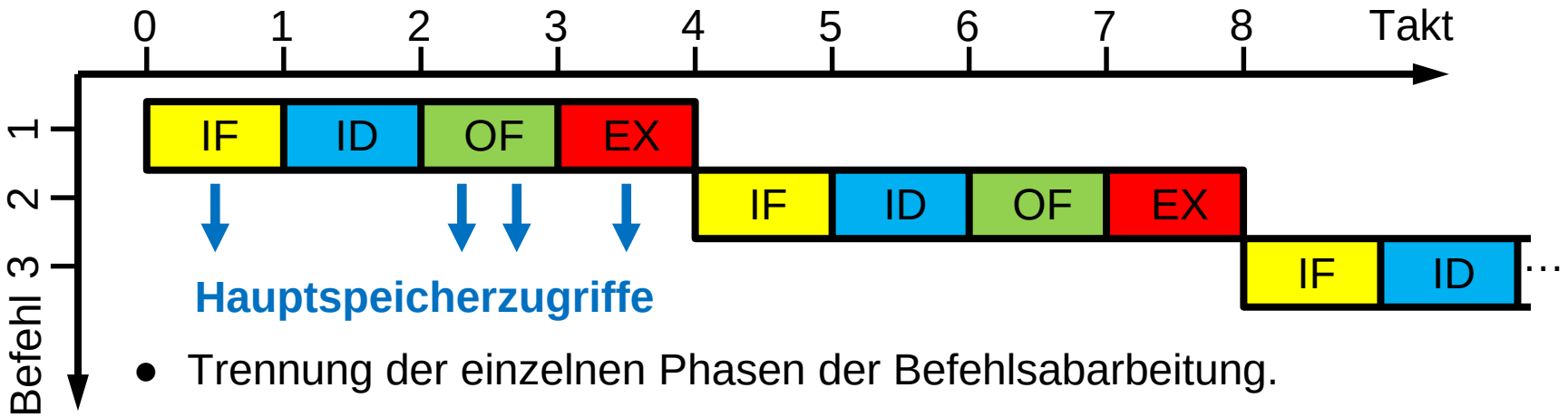


- Konflikte beim Hauptspeicherzugriff, vier Zugriffe: Befehl, 1. Operand, 2. Operand, Resultat → nur ein Hauptspeicherzugriff pro Takt möglich!
- Taktperiode entspricht einem vollen Befehlszyklus → nur ca. ein viertel Takt pro Einheit IF, ID, OF, EX → große Taktperiode.
- Keine Parallelität

CPI = 1

Mehrtakt-Befehlsabarbeitung

Viertakt-Befehlsabarbeitung



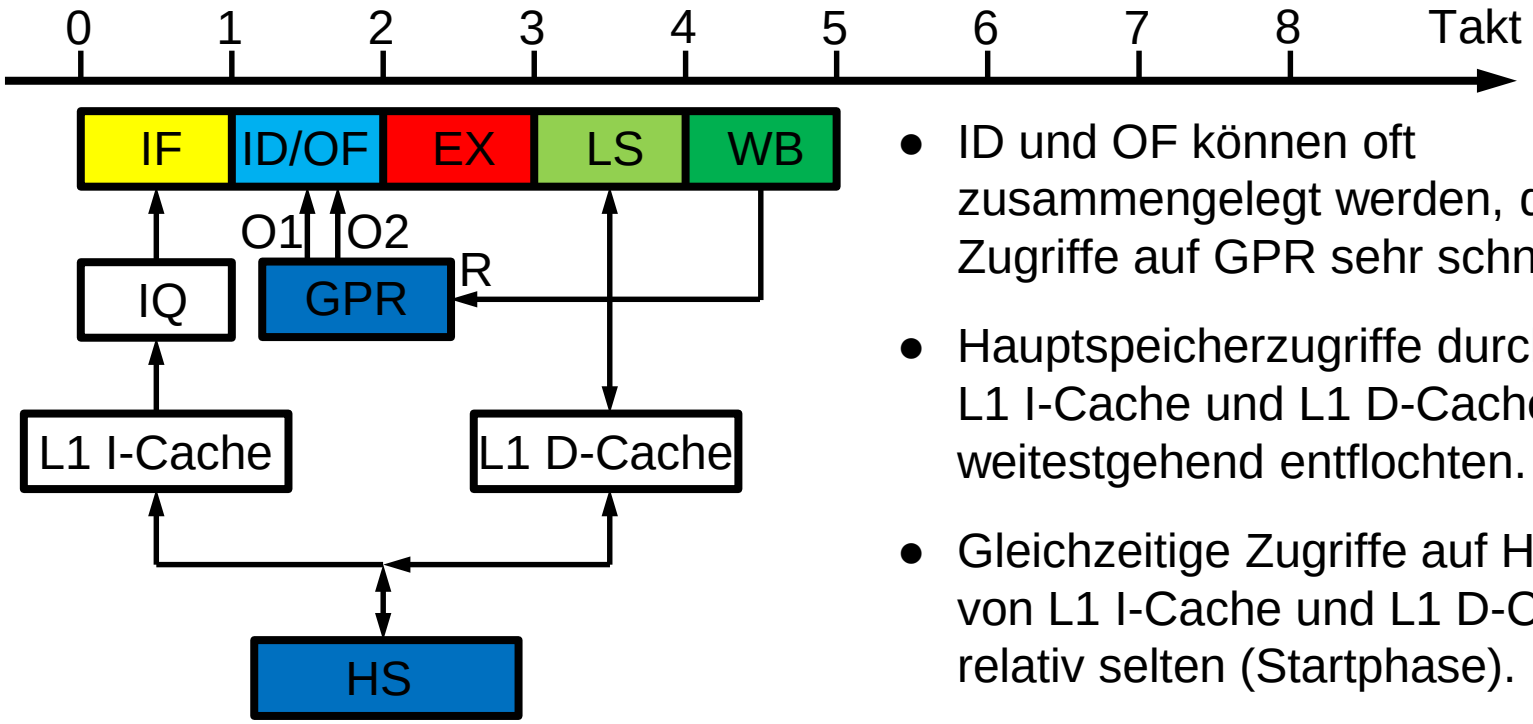
- Trennung der einzelnen Phasen der Befehlsabarbeitung.
- Hauptspeicherzugriffe entflochten (bis auf OF).
- Taktperiode richtet sich nach Phase mit längster Verarbeitungszeit.
- Kürzere Taktperiode (ca. ein viertel) → ca. 4-fach höhere Taktfrequenz.
- Keine Parallelität **CPI = 4** → aber höhere Taktfrequenz.

Entflechtung der Speicherzugriffe

IF-Phase	Zugriff auf Befehle über eine Befehlswarteschlange IQ (Instruction Queue, Instruction Prefetching). IF → IQ → L1 I-Cache → Hauptspeicher
OF-Phase	Zugriffe auf Registersatz, GPR, kein direkter Speicherzugriff. Rückschreiben in GPR in gesonderte WB-Phase- GPR werden in gesonderte Load/Store-Phase, LS, geladen bzw. gespeichert.
LS-Phase	Gesonderte Phase nur für Daten-Speicherzugriffe in einer Load/Store-Architektur (Load/Store, Memory). LS → L1 D-Cache → Hauptspeicher
WB-Phase	Gesonderte Phase nur für das Rückschreiben des Resultates in den Registersatz (Write Back), kein Speicherzugriff.

Mehrtakt-Befehlsabarbeitung

Fünftakt-Befehlsabarbeitung



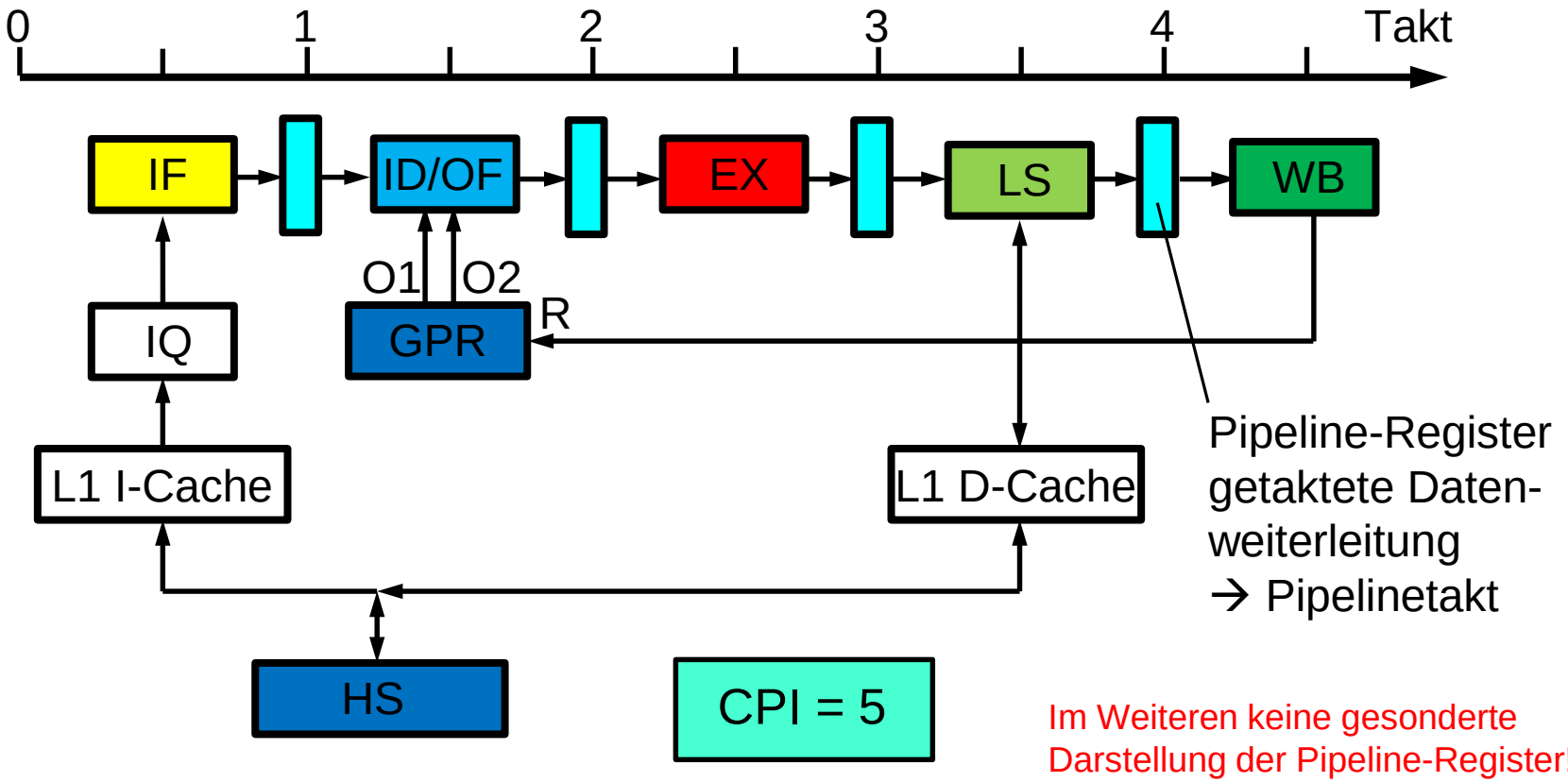
- ID und OF können oft zusammengelegt werden, da Zugriffe auf GPR sehr schnell.
- Hauptspeicherzugriffe durch IQ, L1 I-Cache und L1 D-Cache weitestgehend entflochten.
- Gleichzeitige Zugriffe auf HS von L1 I-Cache und L1 D-Cache relativ selten (Startphase).

CPI = 5

bei ca. 5-facher Taktfrequenz gegenüber Eintaktverarbeitung

RISC-Pipelining

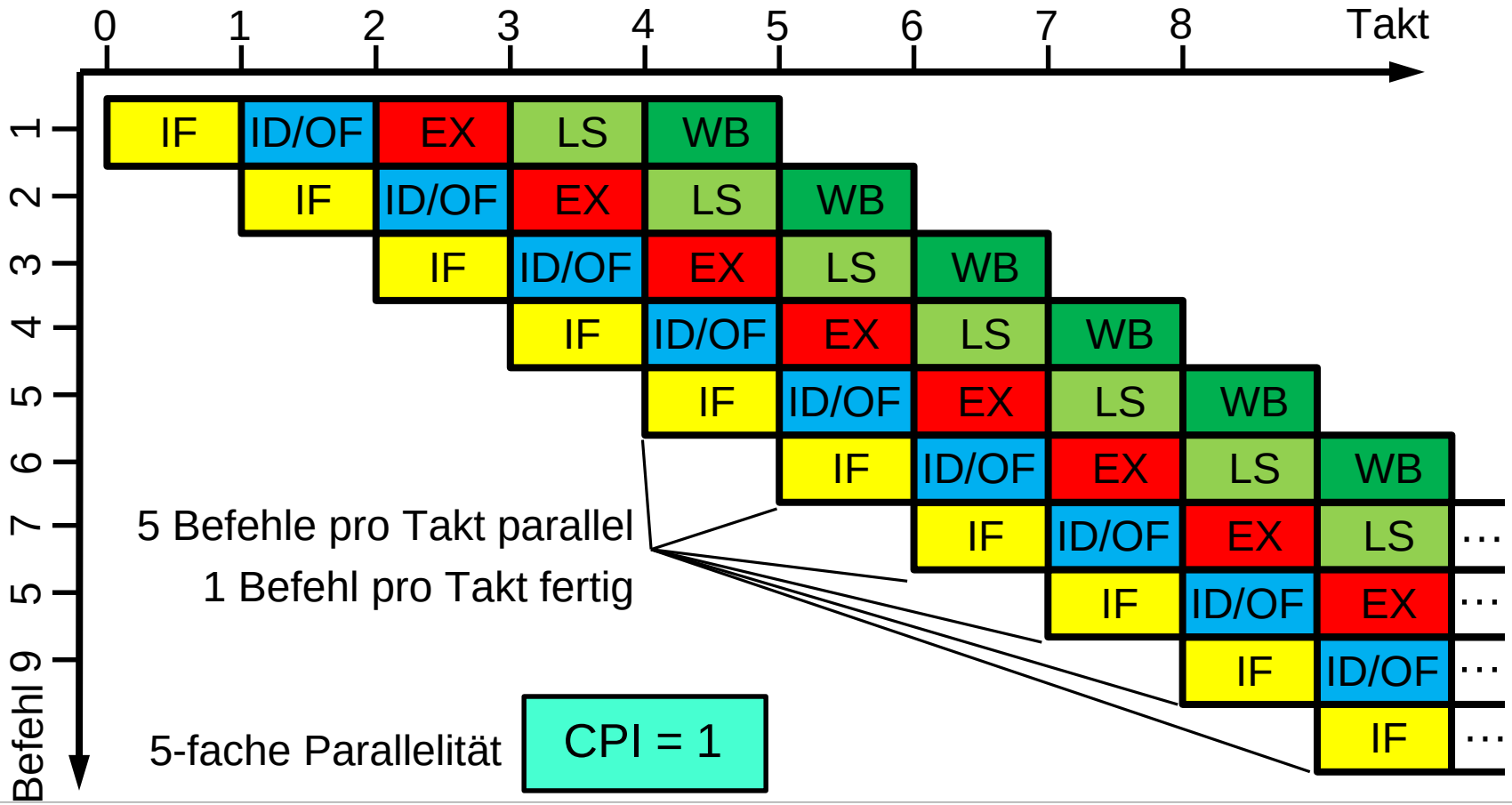
Fünfstufiger Befehlszyklus mit Datenübergabe durch Pipeline-Register



Im Weiteren keine gesonderte Darstellung der Pipeline-Register!

Befehls-Pipelining

Fünftakt-Befehlsabarbeitung – fünfstufige Pipeline



Befehls-Pipelining, Ergebnisse

- Befehle werden um einen Takt zeitversetzt, überlappend gestartet.
- Ab dem 5. Takt wird in jedem Takt ein Befehl fertig → Fließbandprinzip,
- Ab dem 5. Takt arbeiten alle 5 Pipelinestufen parallel, jede an einem anderen Befehl → Befehlsebenenparallelität.
- Startlatenz: 5 Takte. 5 Takte Wartezeit bis Pipeline gefüllt und erstes Ergebnis vorliegt → Laden der Pipeline.
- Nach dem Start des letzten Befehls läuft die Pipeline noch 5 Takte bis der letzte Befehl abgearbeitet ist → Entladen der Pipeline.
- Die Stufenzahl der Pipeline kann beliebig gewählt werden.
- Einzelne Stufen können weiter unterteilt oder zusammengefasst werden.
- Für jede Stufe, Phase der Befehlsabarbeitung steht die selbe Zeit zur Verfügung, eine Taktperiode → die Phasen sollte alle gleich lang sein.

Leistungsbetrachtung zum Pipelining

N - Anzahl der Befehle

S - Anzahl der Pipeline-Stufen

T_C - Taktzykluszeit (einer Stufe)

f_C - Taktfrequenz $f_C = 1/T_C$

Abarbeitungszeit ohne und mit Pipelining

Mehrtaktabarbeitung
seriell, S-Takt-Abarbeitung: TS_{EXE}

$$TS_{EXE} = N \cdot S \cdot T_C$$

Pipeline-Abarbeitung
parallel, S-Stufen-Pipeline: TP_{EXE}

$$TP_{EXE} = (S + N - 1) \cdot T_C$$

Leistungssteigerung durch Pipelining in Abhängigkeit von Stufenzahl und Befehlsanzahl

Speed-Up SP

$$SP = \frac{TS_{EXE}}{TP_{EXE}} = \frac{N \cdot S}{N + S - 1}$$

Effizenz EF

$$EF = \frac{SP}{S} = \frac{N}{N + S - 1}$$

Leistungssteigerung durch 5-stufige Pipeline (S=5)

N	1	2	3	4	5	10	20	50	100	1000
SP	1	1,67	2,14	2,50	2,78	3,57	4,16	4,63	4,81	4,98

Grenzwerte für $N \to \infty$

$$\lim_{N\to\infty} SP = \lim_{N\to\infty} \frac{N \cdot S}{N + S - 1} = S$$

$$\lim_{N\to\infty} EF = \lim_{N\to\infty} \frac{N}{N + S - 1} = 1$$

$$\lim_{N\to\infty} CPI = \lim_{N\to\infty} \frac{TP_{EXE}}{N \cdot T_C} = \lim_{N\to\infty} \frac{N + S - 1}{N} = 1$$

- Die Leistungssteigerung ist direkt von der Stufenzahl S abhängig.
- Die Latenz, Lade- und Endladezeit der Pipeline entspricht $L = S \cdot T_C$.
- Ja kleiner die Stufen-Abarbeitungszeit, je höher die mögliche Taktfrequenz.

Probleme, Konflikte beim Pipelining

Pipeline-Konflikte (Hazards) bzw. Probleme

- Laden und Entladen der Pipeline führt zu zusätzlichen Latenzen.
- Stufenzahl durch Granularität der Befehlsabarbeitung begrenzt.
- Minimale Stufen-Verzögerungszeit durch Pipeline-Register begrenzt (setup).
- Ressourcenkonflikte, Strukturkonflikte (z.B. Speicherzugriffe).
- Datenkonflikte (Datenabhängigkeiten).
- Kontrollflusskonflikte (Programmverzweigungen).

Wirkungen der Konflikte und Probleme

- Pipeline wird nicht optimal ausgelastet, der Durchsatz sinkt.
- Effektive Werte: $SP < S$, $EF < 1$ und $CPI > 1$.
- Zusätzlicher Aufwand zur Konfliktvermeidung und Problembehandlung.

Strukturkonflikte (Structural Hazard)

Ursachen

- Mehrere Stufen wollen gleichzeitig auf eine Ressource zugreifen.
- Ressourcenzugriffe vorgelagerter Befehle sind noch nicht abgeschlossen.
- Unterschiedliche Ressourcen behindern sich gegenseitig.

Wirkung

➔ Ressourcenzugriff nicht eindeutig möglich.

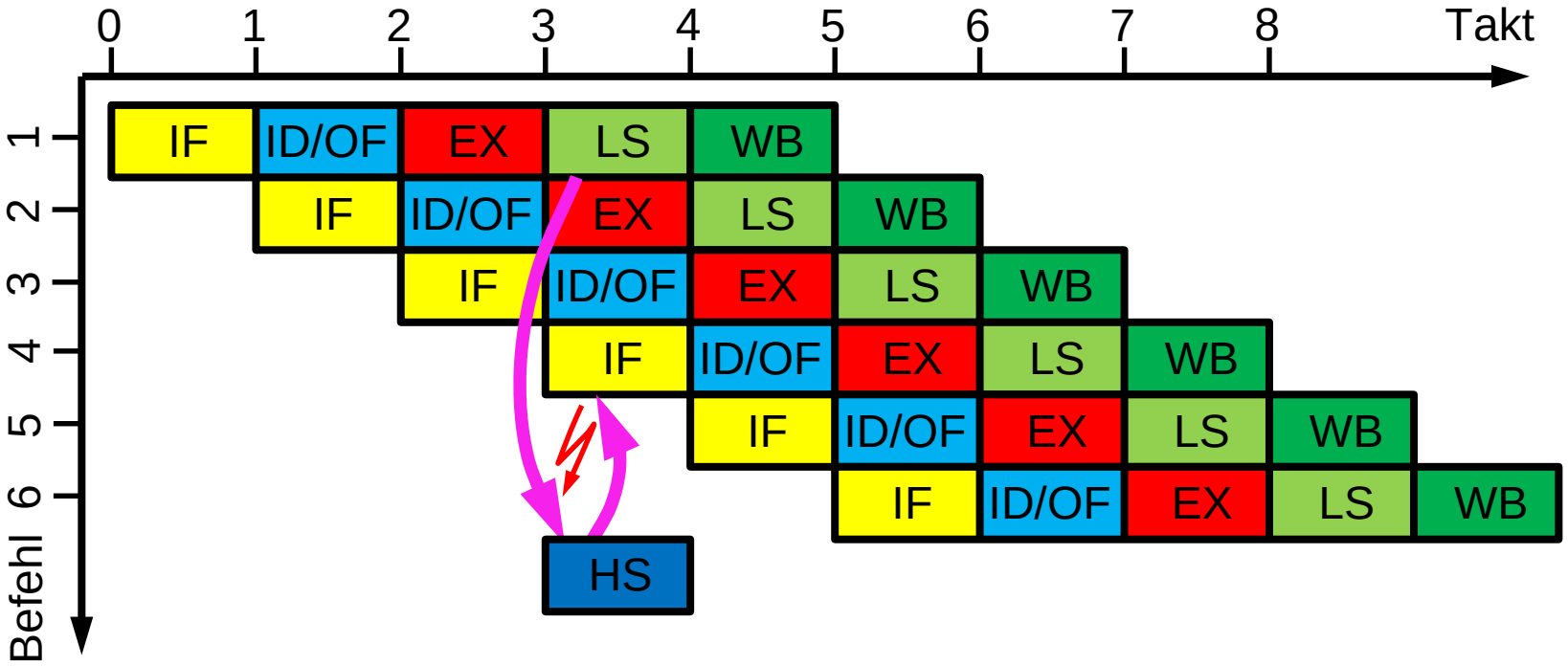
Vermeidung

➔ Entflechtung der Speicherzugriffe, Caches, Registersat, Prefetch Buffer, IQ.

➔ Out-of-Order Execution, Vorziehen, Verlagern von Speicherzugriffen.

➔ Andere Konfiguration der Pipeline-Stufen.

Gleichzeitiger Hauptspeicherzugriff von der IF und der LS Stufe



- Die Pipeline wird angehalten oder der 4. und folgende Befehle gestoppt.
- Der am weitesten fortgeschrittene Befehl wird zuerst bearbeitet, 1. Befehl.
- Pipeline wird wieder gestartet, Bearbeitung fortgesetzt.

Datenkonflikte – Data Hazard, Read after Write (RAW)

Ursachen

- Im Befehl benötigte Registerinhalte sind vom Ergebnis eines vorgelagerten Befehls abhängig, der sich jedoch noch in der Pipeline befindet (vor WB).
- In einer Stufe benötigte Daten stehen noch nicht zur Verfügung (z.B. nach Speicherzugriff).

Wirkung

➔ Verarbeitung veralteter, nicht aktueller Daten.

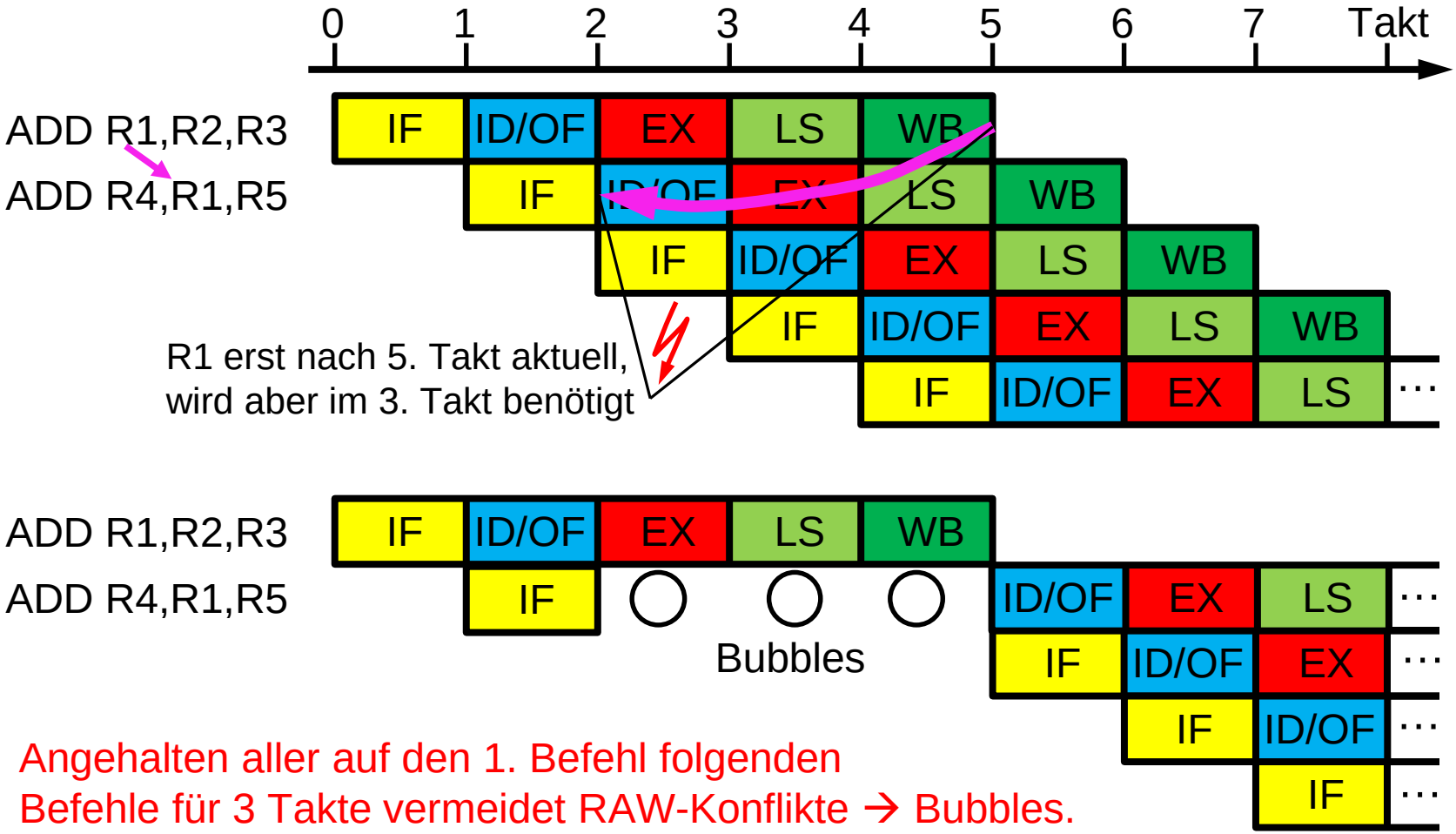
Vermeidung

➔ Anhalten der Pipeline (Bubbles) oder Einfügen von NOP-Befehlen.

➔ Umsortieren der Befehlsfolge durch Nutzer oder Compiler.

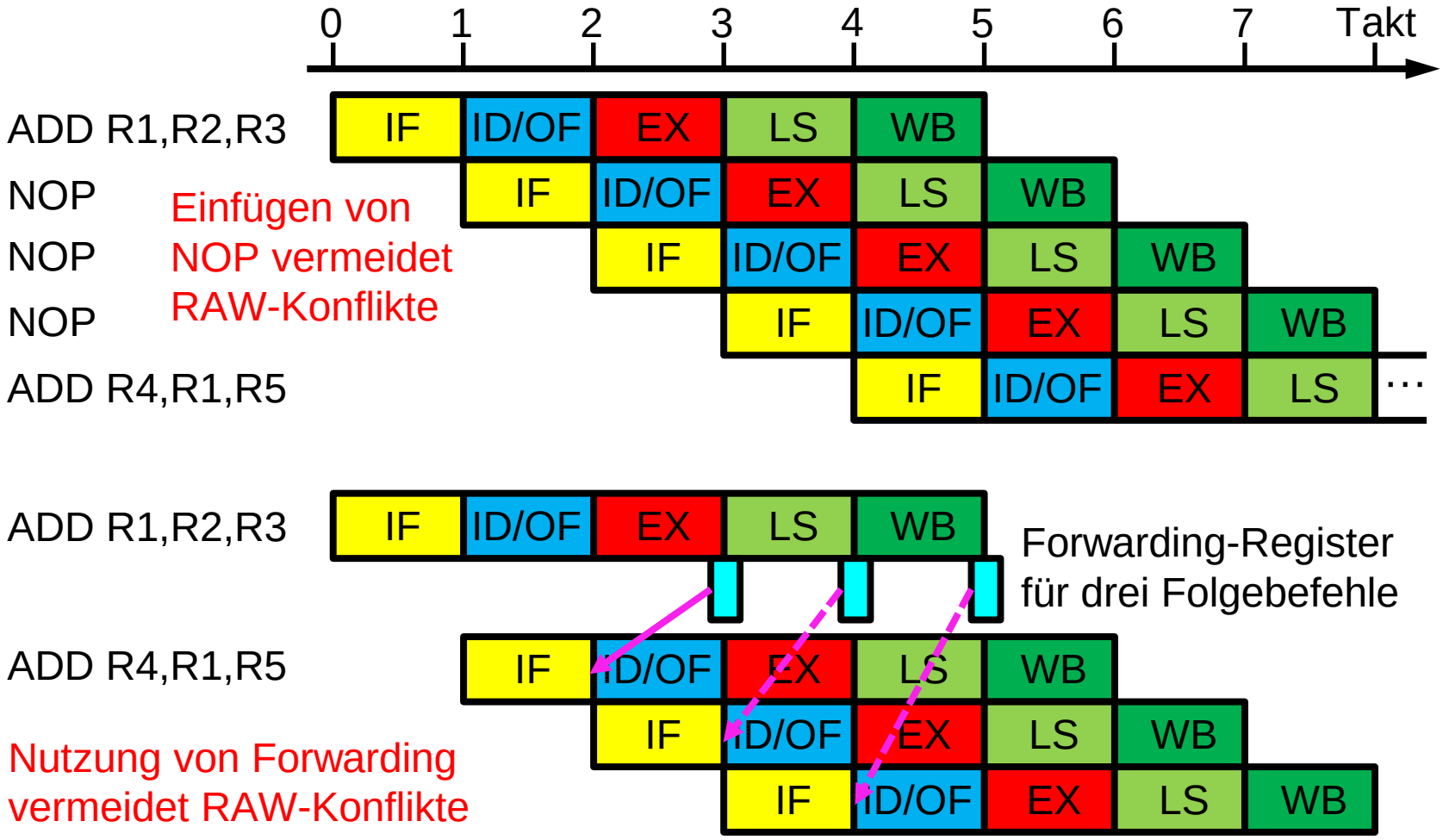
➔ Forwarding, das Ergebnis der Operation liegt bereits nach EX, LS und WB in den Forwarding-Registern vor und kann nach vorn propagiert werden.

Datenkonflikte, Read after Write (RAW)



Angehalten aller auf den 1. Befehl folgenden
Befehle für 3 Takte vermeidet RAW-Konflikte → Bubbles.

Datenkonflikte, Read after Write (RAW)



Umsortieren der Befehlsfolge zur Vermeidung von RAW-Konflikten

Ein Prozessor mit 5-stufiger Pipeline (IF, ID/OF, EX, LS, WB) verfügt nicht über Bypassing oder Forwarding. Wie kann der folgende Befehlsstrom umgestellt werden, um Datenkonflikte zu vermeiden – ggf. durch das Hinzufügen von NOP-Befehlen.

Nr.		Assembler	Kommentar
1.		ADD R4, R3, 10	R4 ← R3 + 10
2.		ADD R1, R3, R3	R1 ← R3 + R3
3.	M1:	LD R6, [R2]	R6 ← Mem[R2]
4.		ADD R1, R1, R6	R1 ← R1 + R6
5.		ADD R2, R2, 4	R2 ← R2 + 4
6.		SUB R4, R4, 1	R4 ← R4 - 1
7.		BNE R4, R3, M1	verzweige zu M1, wenn R3 ≠ R4
...		...	...

Mögliche Lösung

```
        ADD R4, R3, 10
        ADD R1, R3, R3
        NOP
M1:     LD R6, [R2]
        SUB R4, R4, 1
        ADD R2, R2, 4
        NOP
        ADD R1, R1, R6
        BNE R4, R3, M1
        ...
```

Datenkonflikte – Data Hazard, Write after Read (WAR)

Gegenabhängigkeit (Anti-Dependence)

Ursachen

- Im Befehl wird ein Registerinhalt überschrieben, auf den in einem vorhergehenden Befehl lesend zugegriffen wird.

Wirkung

➔ Ein Konflikt tritt immer dann auf, wenn die aktuelle Schreiboperation schneller abgeschlossen ist als die vorhergehende Leseoperation.

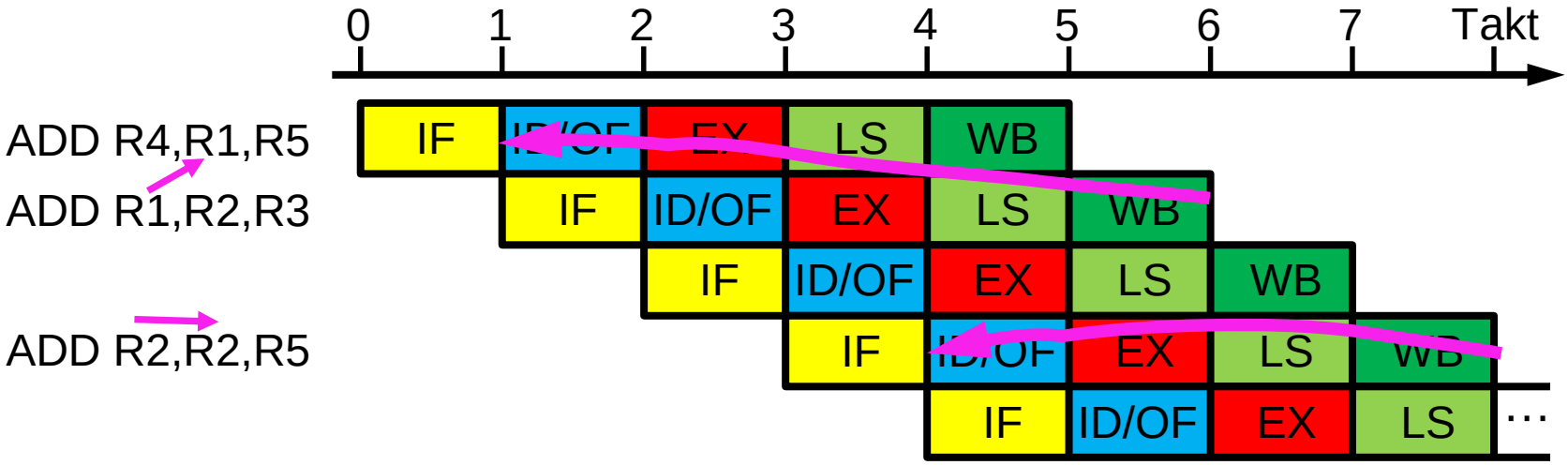
Vermeidung

➔ Stellt bei der realen Pipeline kein Problem, echten Konflikt dar.

➔ Keine Konfliktbehandlung erforderlich.

➔ Beim Umsortieren der Befehlsfolge (z.B. für Lösung des RAW-Konflikt) unbedingt zu beachten.

Datenkonflikte, Write after Read (WAR)



Kein echter Konflikt, WB frühestens 4 Takte später.

Datenkonflikte – Data Hazard, Write after Write (WAW)

Gegenabhängigkeit (Anti-Dependence)

Ursachen

- Im Befehl wird ein Registerinhalt überschrieben, auf den in einem vorhergehenden Befehl ebenfalls geschrieben wird.

Wirkung

➔ Ein Konflikt tritt immer dann auf, wenn die aktuelle Schreiboperation schneller abgeschlossen ist als die vorhergehende Schreiboperation.

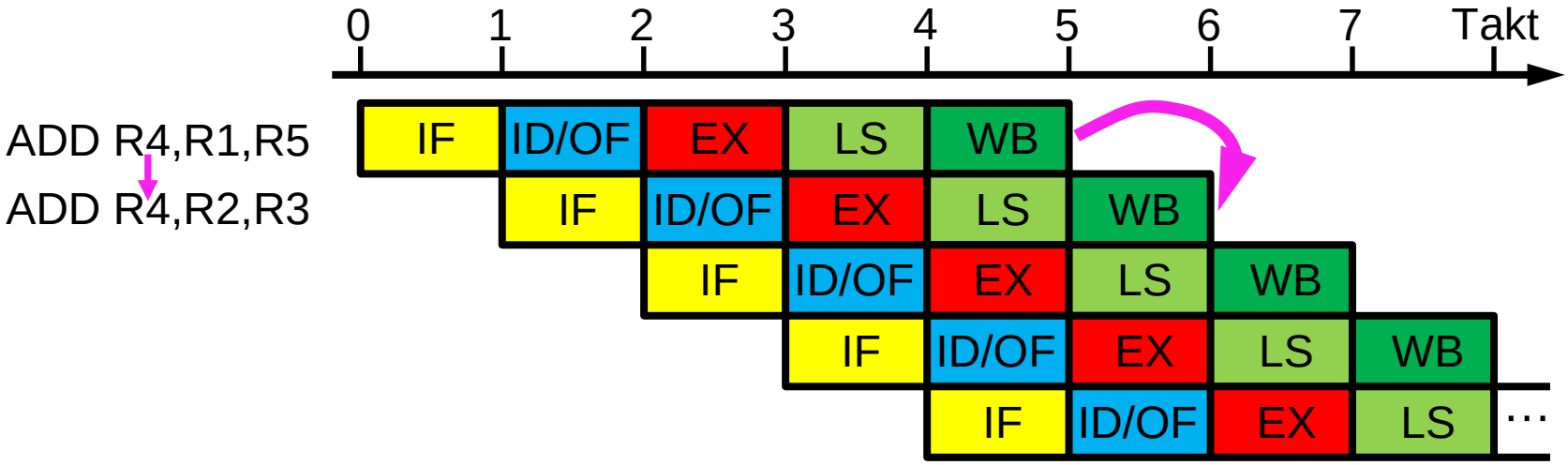
Vermeidung

➔ Stellt bei der realen Pipeline kein Problem, echten Konflikt dar.

➔ Keine Konfliktbehandlung erforderlich.

➔ Beim Umsortieren der Befehlsfolge zu beachten.

Datenkonflikte, Write after Write (WAW)



Kein echter Konflikt, WB frühestens 1 Takte später.

Steuerkonflikte (Control Hazard)

Ursachen

- Verzweigungsbefehle können den sequentiellen Befehlsablauf unterbrechen.
- Bei bedingten Verzweigungen steht erst relativ spät fest, ob verzweigt wird oder nicht. Das trifft auch auf das Sprungziel zu.

Wirkung

➔ Befehle nach Verzweigungsbefehlen werden in der Pipeline abgearbeitet, obwohl sie eigentlich übersprungen werden sollten.

Vermeidung

➔ Anhalten der Pipeline (Pipeline Stall), Einfügen von NOP-Befehlen.

➔ Umsortieren der Befehlsfolge, Out-of-Order Execution.

➔ Sprungvorhersage, spekulative Befehlsabarbeitung.

Beispiel: Bedingt Verzweigung

Zweistufige Realisierung

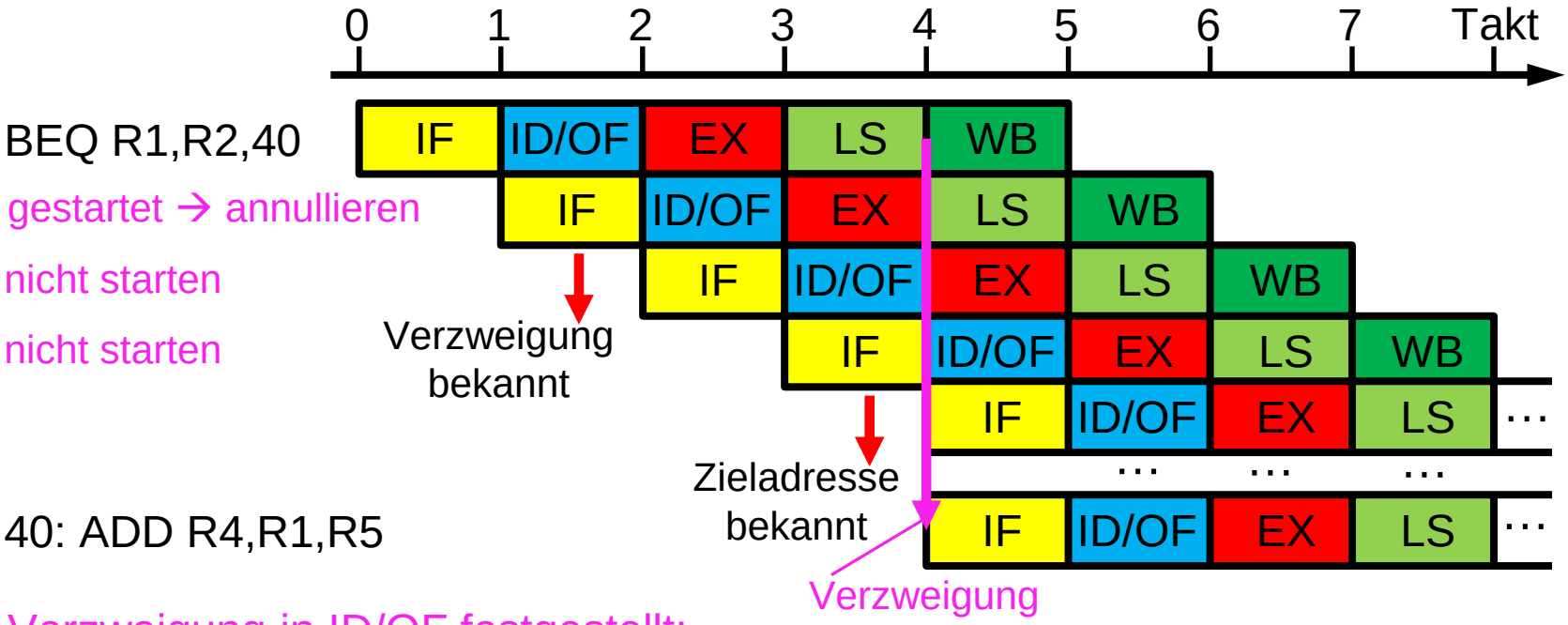
1. Ergebnisse einer arithmetischen Operation sind auch Flags, aus denen der Bedingungscode (Condition Code, cc) berechnet werden kann.
2. Bestimmung, Berechnung der Zieladresse in Abhängigkeit vom Bedingungscode, den Flags der letzten arithmetischen Operation und setzen des Befehlszählers auf die Sprungadresse.

Kombination von arithmetischer Operation und Verzweigungsoperation in einem Befehl sind auch möglich.

Randbedingungen (Beispiel)

- Der Bedingungscode liegt erst nach der EX-Stufe vor.
- Verzweigungsbefehl wird erst in der ID-Stufe identifiziert.
- Der Befehlszähler (Program Counter) wird in der LS-Stufe gesetzt.

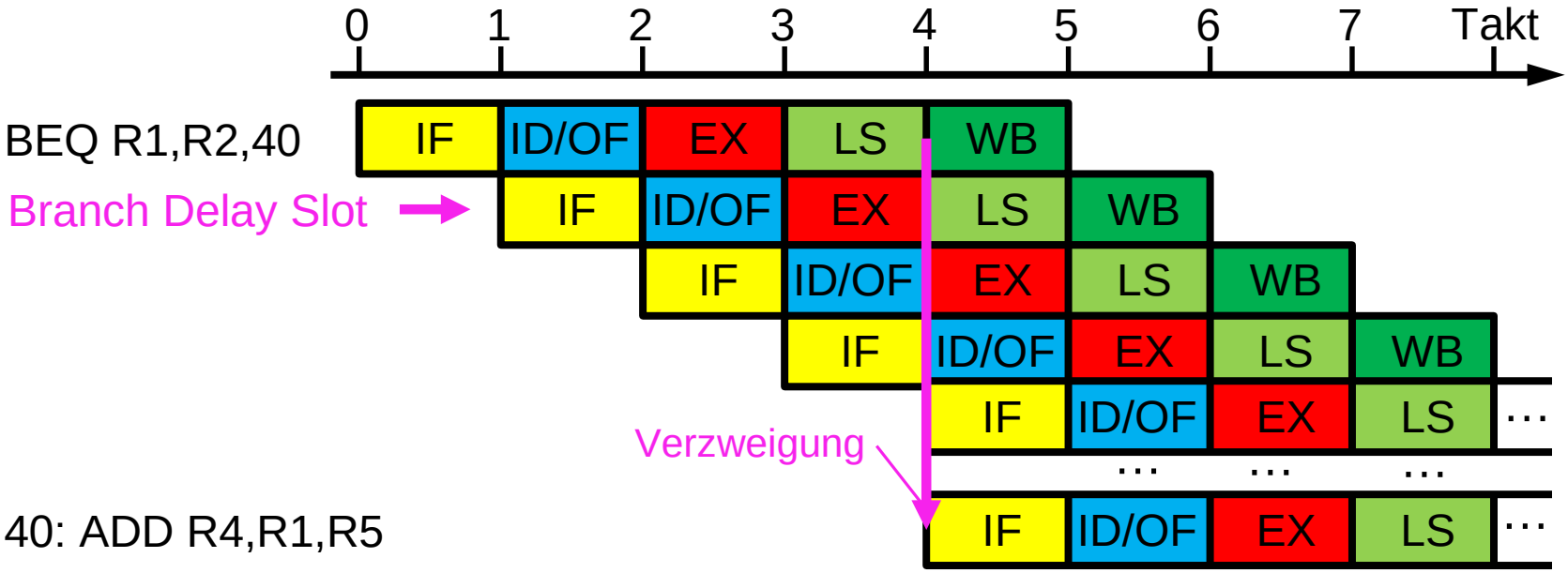
Seuerkonflikte (Control Hazard)



Verzweigung in ID/OF festgestellt:

- 2 Befehl bereits gestartet → Befehl muss annulliert werden.
- 3. und 4. Befehl darf nicht mehr gestartet werden bzw. NOP dafür gestartet.
- Verzweigung entsprechend Verzweigungsadresse.

Seuerkonflikte (Control Hazard) – Branch Delay Slot



Branch Delay Slot: Folgebefehl auf bedingte Verzweigung.
Befehl wird immer ausgeführt, auch bei Verzweigung.
Umsortieren der Befehlsfolge, wenn nicht möglich mit NOP.

2., 3. Folgebefehl nach bedingter Verzweigung bei Verzweigung nicht gestartet.

Einfachste Konfliktvermeidung wäre 3 NOP nach bedingter Verzweigung.

Vermeidung, Reduktion von Steuerkonflikten

Steuerkonflikte beeinflussen Durchsatz und damit die Leistungsfähigkeit einer Pipeline wesentlich → Vermeidung bzw. Reduktion von Steuerkonflikten.

Maßnahmen zur Reduktion von Steuerkonflikten

1. Allgemeine Vermeidung von Verzweigungsbefehlen:
   - Aufrollen von Schleifen (Loop Unrolling).
   - Direktes Einfügen von Unterprogrammen.
   - Spezialbefehle, die in Abhängigkeit einer Bedingung ausgeführt werden.
2. Änderungen der Architektur:
   - Vorverlagerung der Sprungentscheidung (z.B. in die EX-Stufe).
   - Vorverlagerung der Sprungadressenberechnung (z.B. in die ID-Stufe). → Look-Ahead-Resolution.
   - Sonderbehandlung unbedingter Verzweigungen.

3. Spekulative Befehlsausführung:
   - Spekulative Befehlsabarbeitung mit einer Vorzugsverzweigung.
   - Spekulative Befehlsabarbeitung beider Verzweigungsrichtungen.
4. Verzweigungsbefehle mit Branch Delay Slot:
   - Befehlsfolge umsortieren, einfügen verzweigungsunabhängiger Befehle.
   - Spekulatives Einfügen der Befehle der Vorzugsverzweigung.
5. Verzweigungsvorhersage (Branch Prediction):
   - Statische Verzweigungsvorhersage (Verzweigungsstatistik, Compiler).
   - Dynamische Verzweigungsvorhersage:
     Verzweigungs-Mustererkennung
     Verzweigungssammlung (Branch-History-Table),
     Verzweigungsmuster (Pattern-History-Table),
     Verzweigungsadressen (Branch-Target-Buffer, Prediction-Cache).

Umsortieren der Befehlsfolge zur Vermeidung von Steuerkonflikten

Ein Prozessor mit 5-stufiger Pipeline verfügt über Forwarding und einen Branch Delay Slot. Wie kann der folgende Befehlsstrom umgestellt werden, um Steuerkonflikte zu vermeiden – ggf. durch das Hinzufügen von NOP-Befehlen.

Nr		Assembler	Kommentar
1.		ADD R4, R3, 10	R4 ← R3 + 10
2.	M1:	LD R6, [R2]	R6 ← Mem[R2]
3.		ADD R1, R1, R6	R1 ← R1 + R6
4.		BEQ R1, R4, M2	verzweige zu M2, wenn R1 = R4
5.		ADD R2, R2, 4	R2 ← R2 + 4
6.		SUB R4, R4, 1	R4 ← R4 - 1
7.		BNE R4, R3, M1	verzweige zu M1, wenn R0 ≠ R4
8.	M2:	ADD R1, R3, R3	R1 ← R3 + R3

Mögliche Lösung

```
        ADD R4, R3, 10
M1:     LD R6, [R2]
        ADD R1, R1, R6
        BEQ R1, R4, M2
        NOP
        SUB R4, R4, 1
        BNE R4, R3, M1
        ADD R2, R2, 4
M2:     ADD R1, R3, R3
…
```

5.2 Daten-Pipelining

Pipelining wird ebenfalls zur Parallelisierung von Datenpfaden eingesetzt
→ Ausführungs-Pipeline.

Beispiel: mögliche Stufen einer Gleitkomma-Pipeline

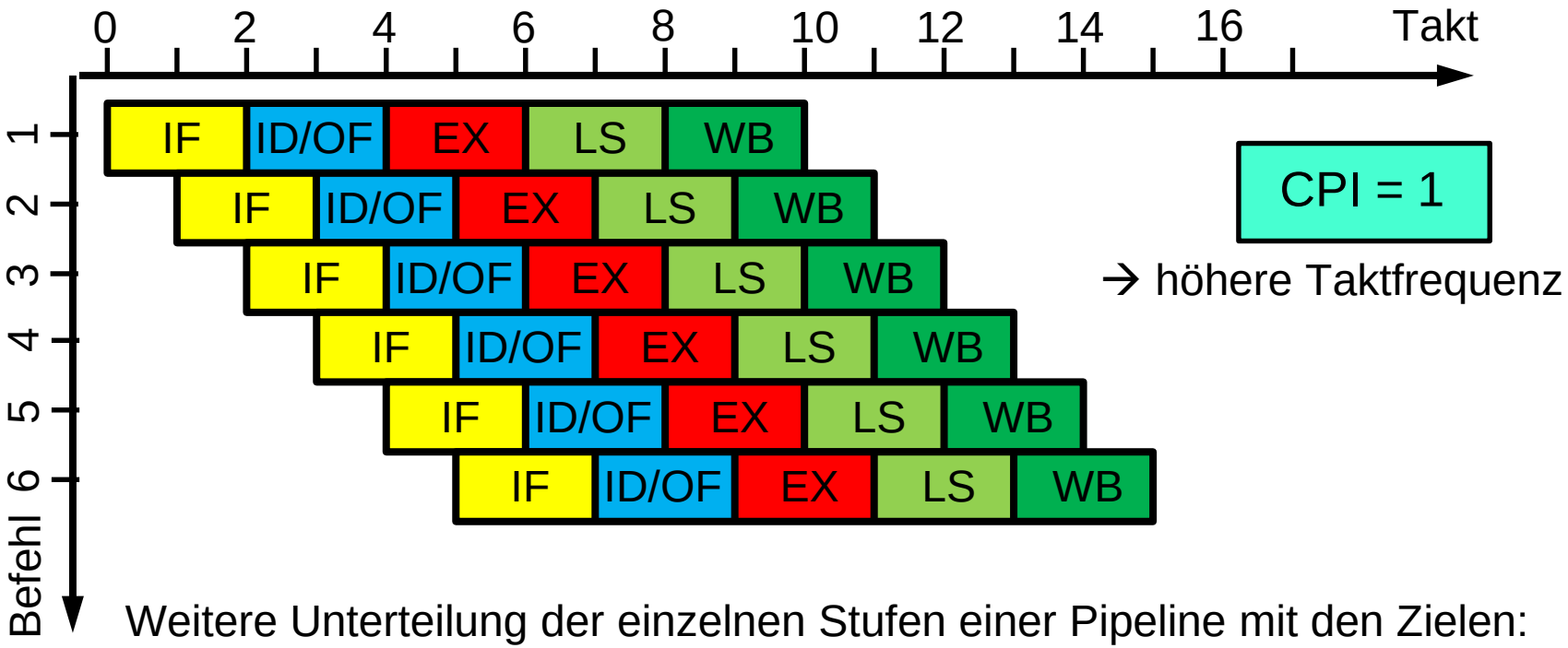
Gleitkomma-Addition	**Gleitkomma-Multiplikation**
1. Exponenten-Subtrahierer	1. Exponenten-Addierer
2. Exponenten-Angleichung	2. Mantissen-Multiplizierer
3. Mantissen-Addierer	3. Rundung
4. Normalisierer	4. Normalisierer

Beispiel: Divisions-Pipeline, numerische Verfahren

Beispiel: CORDIC-Pipeline (Coordinate Rotation Digital Computer)

5.3 Super-Pipelining

10-stufige RISC-Pipeline



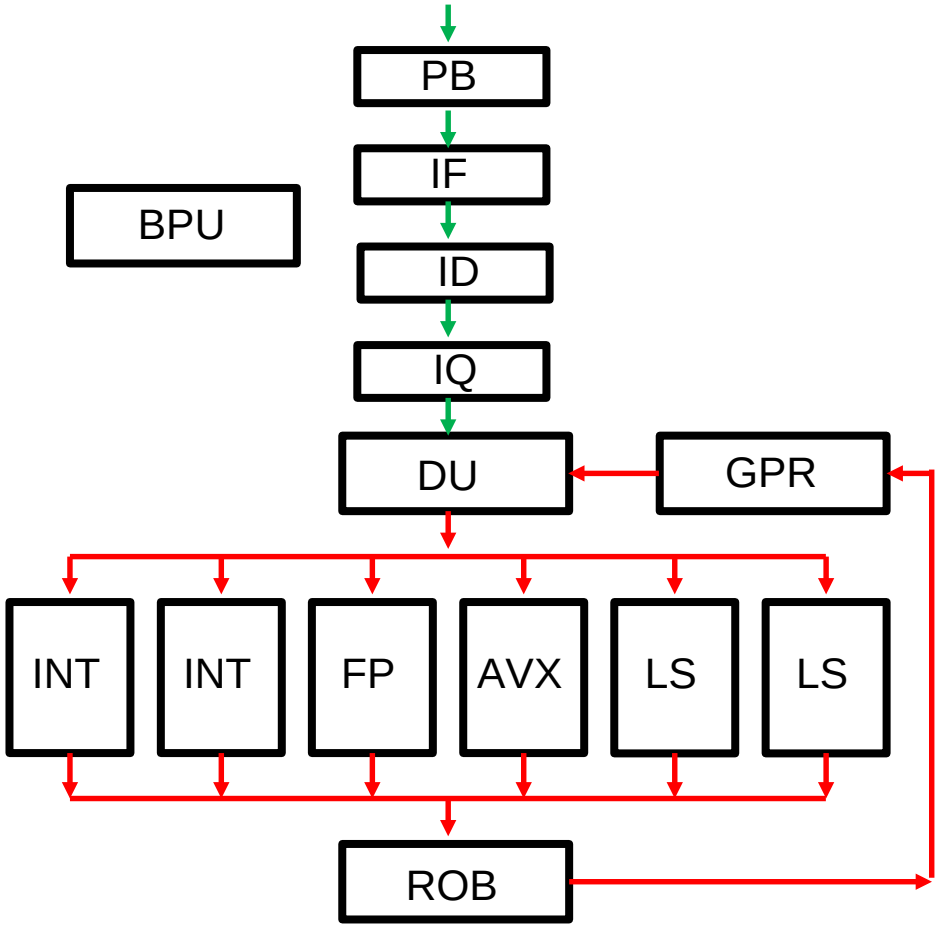
Weitere Unterteilung der einzelnen Stufen einer Pipeline mit den Zielen:

- Feinere Abstimmung der Pipeline, bessere Stufenauslastung.
- Weitere Erhöhung der Taktfrequenz und des Pipeline-Durchsatzes.

5.4 Superskalar-Pipelining

- Innerhalb der Pipeline werden mehrere Verarbeitungseinheiten parallel angeordnet (mehrere EX-Einheiten) → mehrere Befehle parallel.
- Die einzelnen Verarbeitungseinheiten sind typisch auf bestimmte Gebiete spezialisiert (INT, FP, LS, . . .)
- Die Hardware verteilt die abzuarbeitenden Befehle entsprechend auf die einzelnen Verarbeitungseinheiten (Dispatch Unit).
- Der Befehlsstrom wird so innerhalb der Pipeline auf mehrere Verarbeitungseinheiten verteilt, parallelisiert.
- Durchlaufzeit der einzelnen Befehle ist unterschiedlich (Reorder Buffer).
- Der Parallelisierungsgrad durch ILP kann um die Anzahl der parallelen Verarbeitungseinheiten erhöht werden (wird praktisch nicht erreicht).
- Die Anzahl der Pipeline-Konflikte nimmt aufgrund der parallelen Abarbeitung mehrerer Befehle innerhalb der EX-Phase stark zu.

Superskalare- Pipeline (stark vereinfacht)

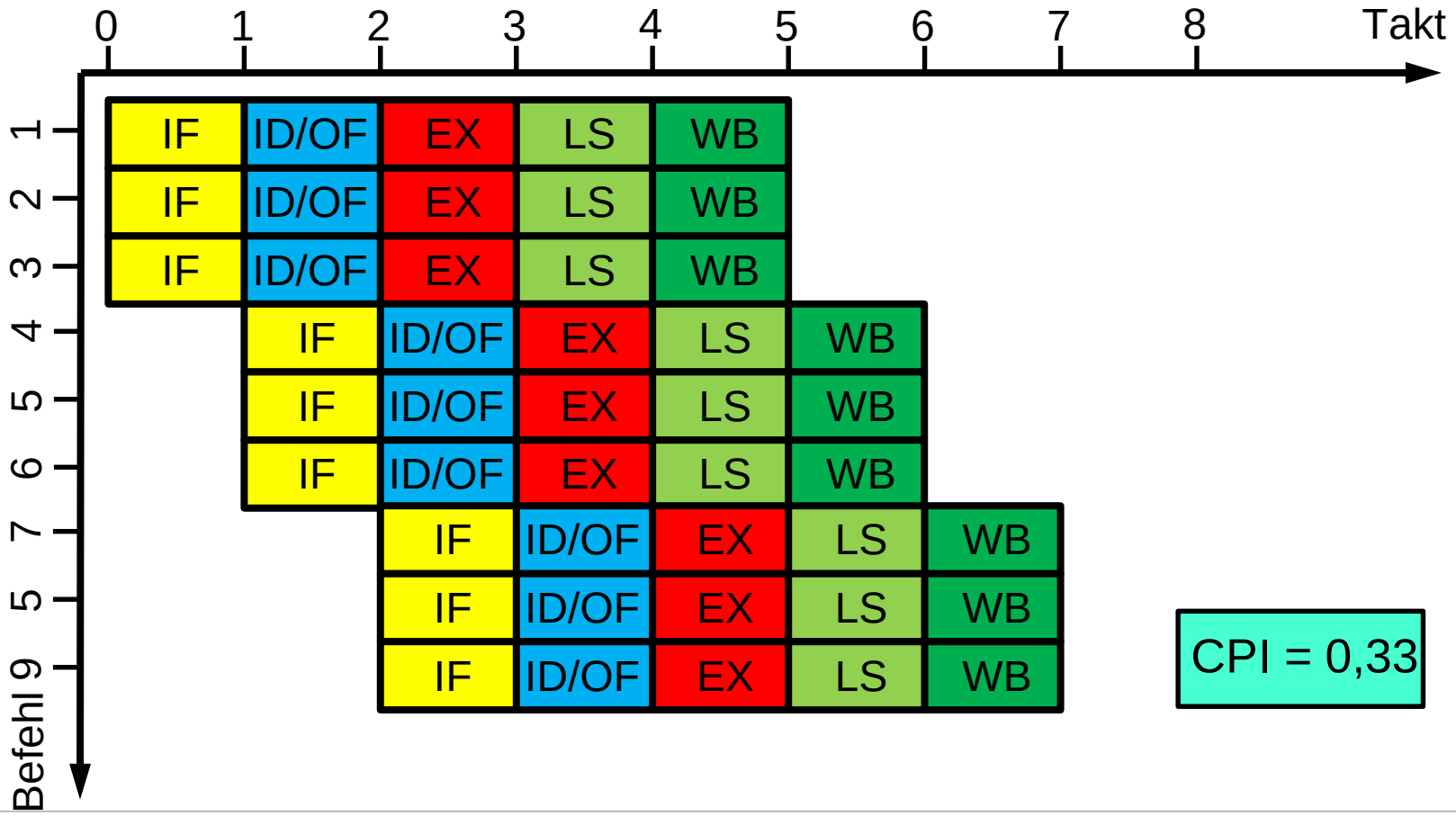


PB	Prefetch Buffer
IF	Instruction Fetch
ID	Instruction Decode
IQ	Instruction Queue
DU	Dispatch Unit
GPR	General Purpose Register
INT	Integer Pileline
FP	Floating Point Pipeline
LS	Load/Store Unit
AVX	Advanced Vector Extensions (MMX / 3DNow!, SSE)
ROB	Reorder Buffer
BPU	Branch Prediction Unit

In Order Execution oder
Out of Order Execution

Superskalares-Pipelining

5-stufige RISC Pipeline, 3-fach superskalar



5.5 Multitheading

Multithreading (MT):

Mehrfädigkeit, gleichzeitiges bzw. quasi-gleichzeitiges Abarbeiten mehrerer Threads (Ausführungsstränge) innerhalb eines Prozesses oder Tasks (Ausführungsprogramms). Unterschied zu Multiprocessing (echt parallele Ausführung durch mehrere Prozessorkerne).

Hardware Multithreading:

Prozessor-Hardware, die von einem Thread gerade nicht verwendet wird, kann von einem anderen Thread verwendet werden. Nur marginaler Overhead an CPU-Ressourcen. Die meisten werden von allen Threads des Prozessors gleichermaßen verwendet.

Simultaneous Multithreading (**SMT)**:

Fähigkeit eines Prozessors, mittels getrennter Pipelines und/oder zusätzlicher Registersätze mehrere Threads gleichzeitig auszuführen. SMT stellt eine Form des Hardware Multithreadings dar.
z.B. Intels Hyper-Threading-Technik (HTT)

Simultaneous Multithreading (SMT)

Ziel von SMT ist es, die bereits aufgrund der Pipeline-Architektur redundant vorhandenen Ressourcen eines Prozessors noch besser auszulasten. Die Pipeline-Architektur kann nur Befehle innerhalb eines Threads abarbeiten und diese auch nur diese paralellisieren.

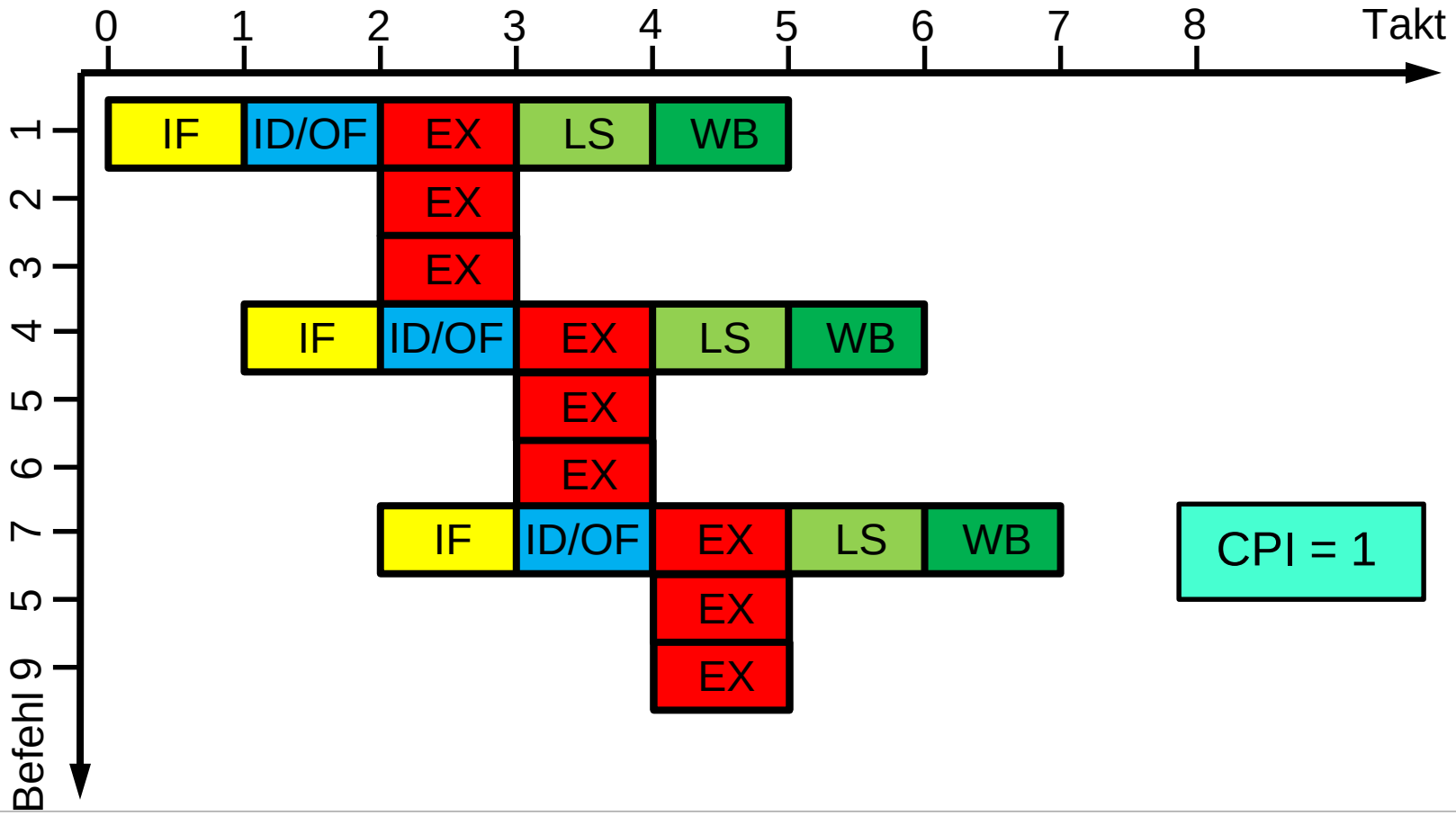
Bei SMT ist die Ausführung mehrerer Threads innerhalb der Pipeline-Architektur gleichzeitig möglich. Dazu werden die Verarbeitungseinheiten der Pipeline erweitert, falls noch nicht wie beim Superskalar-Pipelining bereits vorhanden. Zusätzlich sind noch der Registersatz und die Befehlsdecodierung entsprechend zu erweitern.

Simultaneous Multithreading stellt eine kostengünstige, wenn auch wesentlich leistungsärmere Alternative zu Multicore-Prozessoren dar. Der zusätzliche Hardware-Aufwand ist jedoch demgegenüber sehr gering.

Gegenüber dem System erscheint eine SMT-CPU meist wie mehrere unabhängige Prozessoren (Kerne). Z.B. Intels Quad-Core, 4 echte Kerne, erscheint mit HTT im System als 8 Kerne.

5.6 VLIW-Architekturen

5-stufige RISC Pipeline, 3 parallele Verarbeitungseinheiten



6 Zusammenfassung

- Parallelisierung innerhalb der CPU ist auf allen Ebenen realisierbar.
- Parallelität auf Bit-, Wort-, Befehls-, Thread-Ebene.
- Die Mittel der Wahl sind SIMD und Pipelining.
- Beim Pipelinig sind Ressourcen-, Daten-, und Steuerkonflikte zu beachten und zu lösen.
- Viele Konflikte können bereits im Vorfeld, bei der Programmierung bzw. Compilierung vermieden oder reduziert werden.
- Die theoretisch möglichen Leistungssteigerungen durch Nutzung von Parallelität sind teilweise erheblich.
- Der Befehlsstrom, die Anzahl der parallelisierbaren Befehle und deren Anordnung hat einen wesentlichen Einfluss auf die praktisch realisierbare Leistungssteigerung.