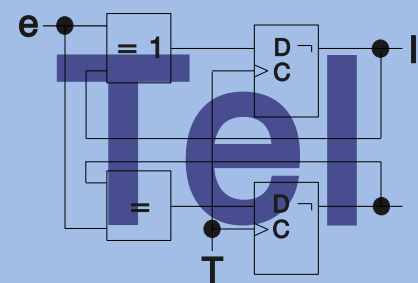


Explicitly Parallel Instruction Computing im IA-64 Itanium Prozessor von Intel

Marcel Wiechmann

Dresden, 25. Juni 2008



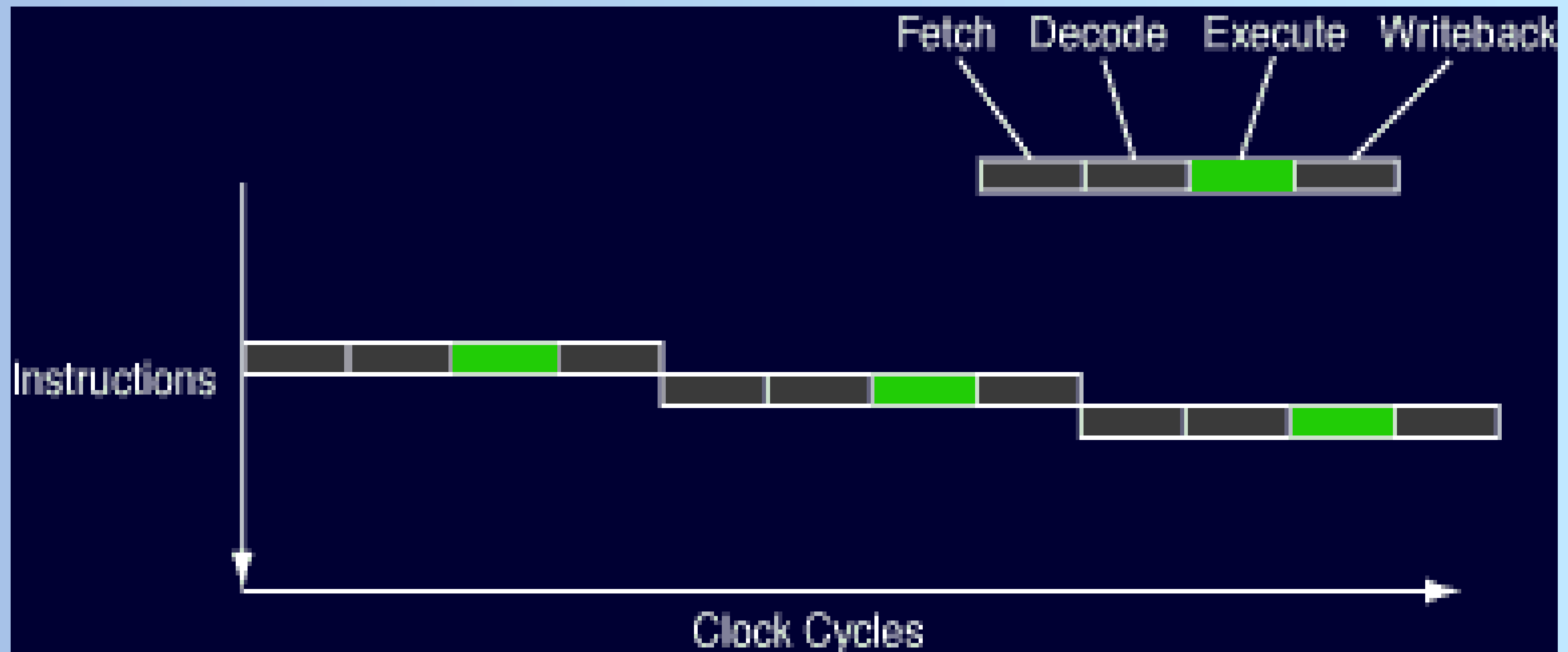
1. Einführung
2. ILP – Architekturen
3. Ausblick

1. Einführung

- CPI / IPC - Rate
- Ausnutzung von ILP
- Optimierung von ILP

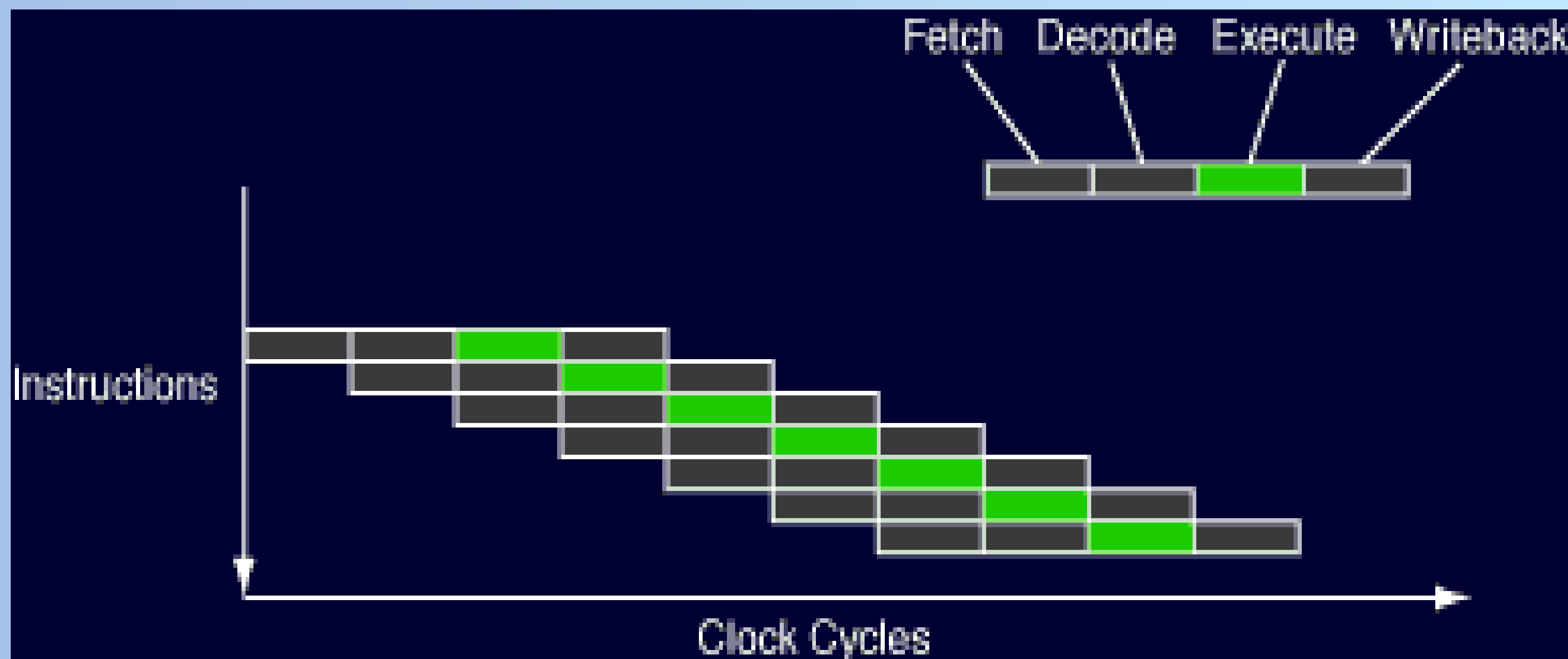
- CPI ... cycles per instruction
- IPC ... instructions per cycle
- $CPI * IPC = 1$
- $IPS = IPC * f$ („brainiac“ vs. „speed demon“ CPU design)
- eingeschränktes Maß für die Effizienz einer Architektur
- feinkörnige Parallelität (zwischen einzelnen Befehlen)
 - zeitlich (Pipelining) $\Rightarrow CPI \rightarrow 1$
 - räumlich (Superskalarität) $\Rightarrow CPI < 1$ ($IPC > 1$)

Kontrollfluss im sequentiellen Prozessor



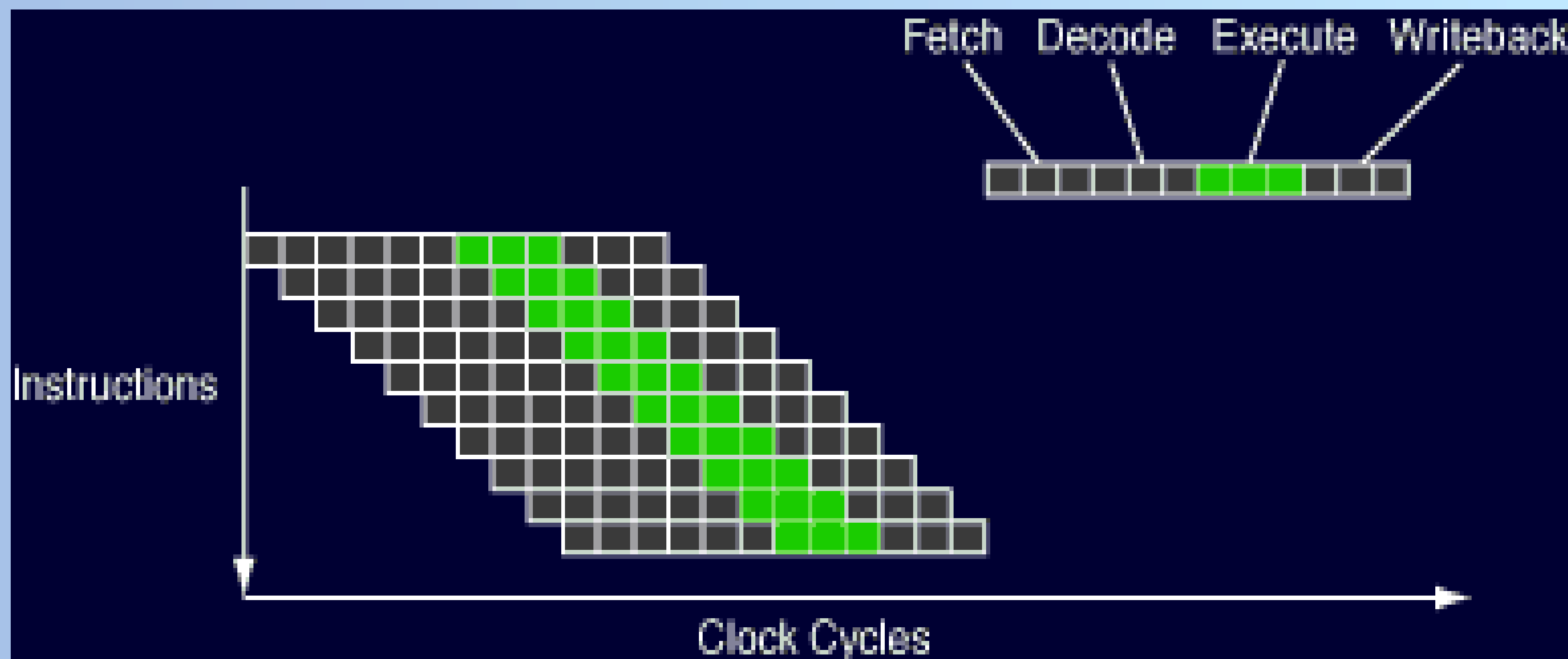
⇒ CPI = 4

Kontrollfluss im Prozessor mit Pipeline



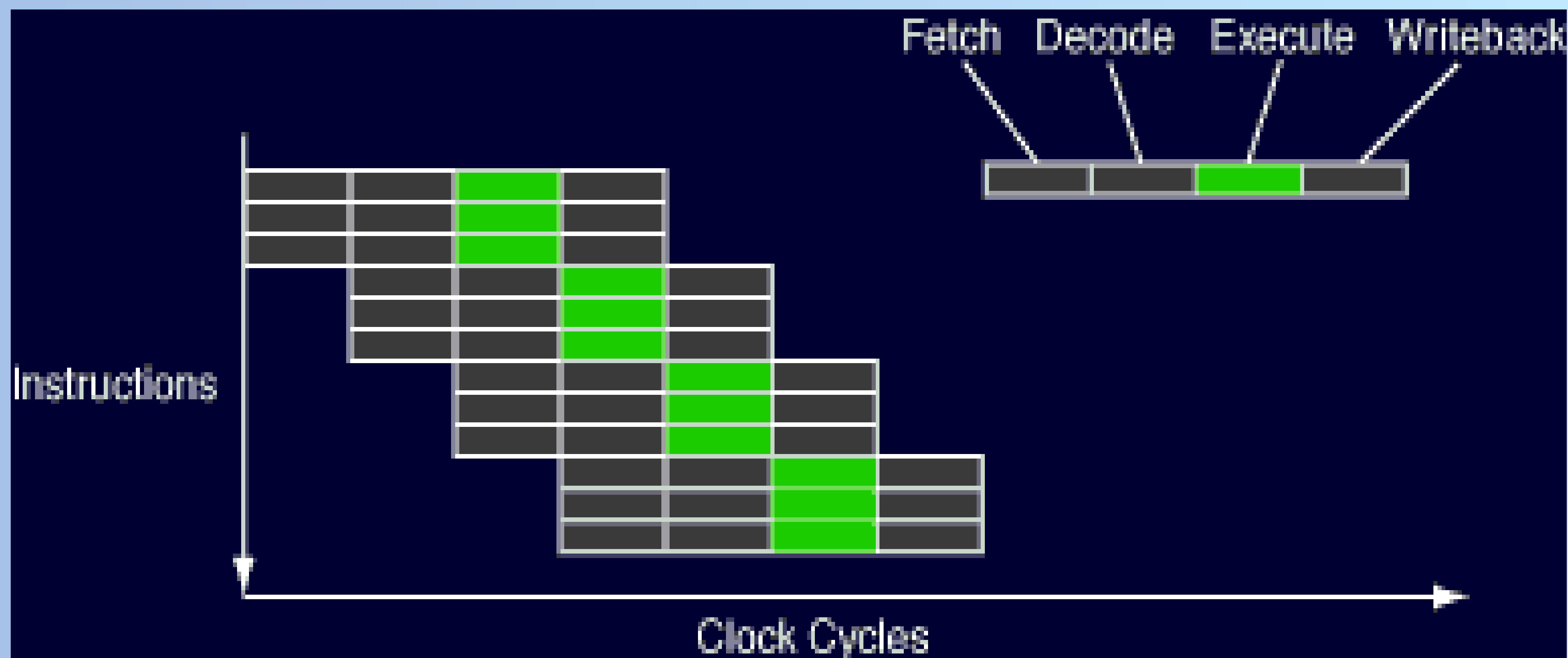
⇒ CPI = 1

Kontrollfluss im Prozessor mit Superpipeline



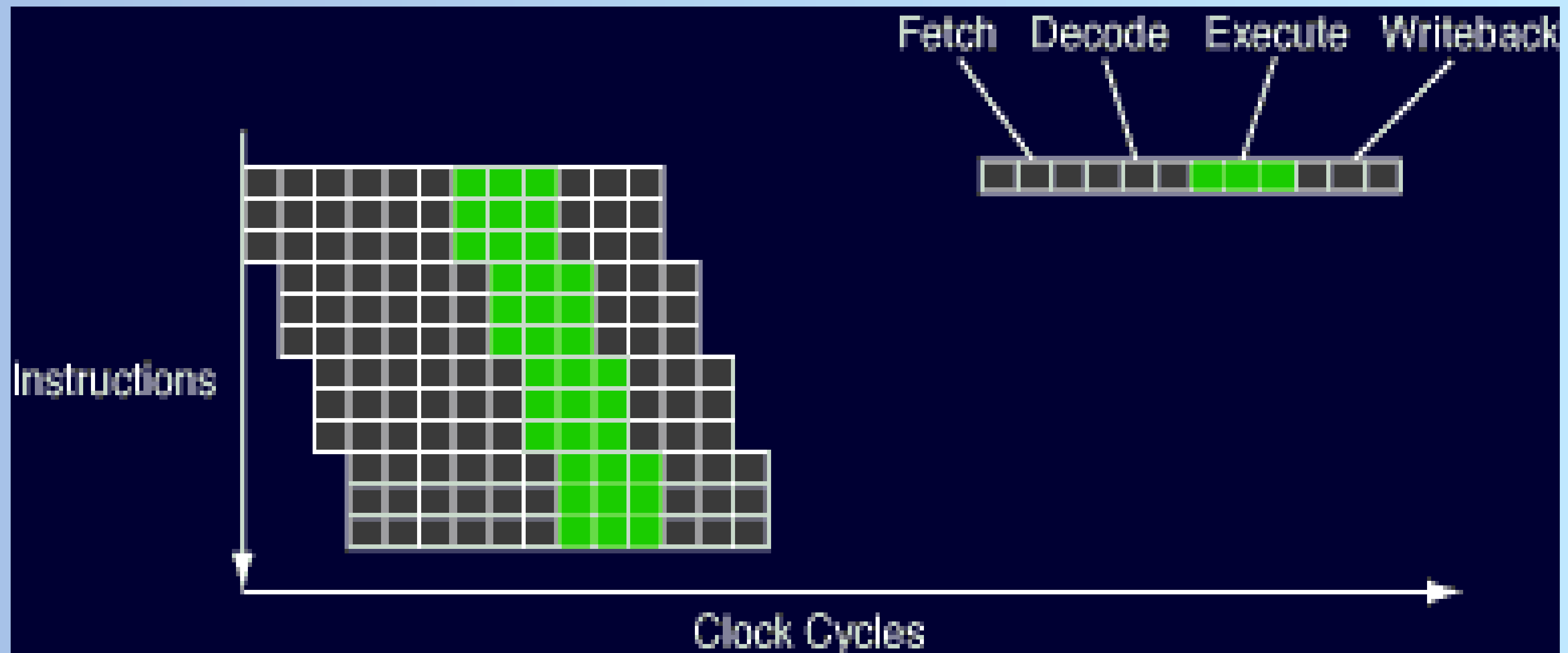
$$\Rightarrow \text{CPI} = 1$$

Kontrollfluss im superskalaren Prozessor (Pipeline)



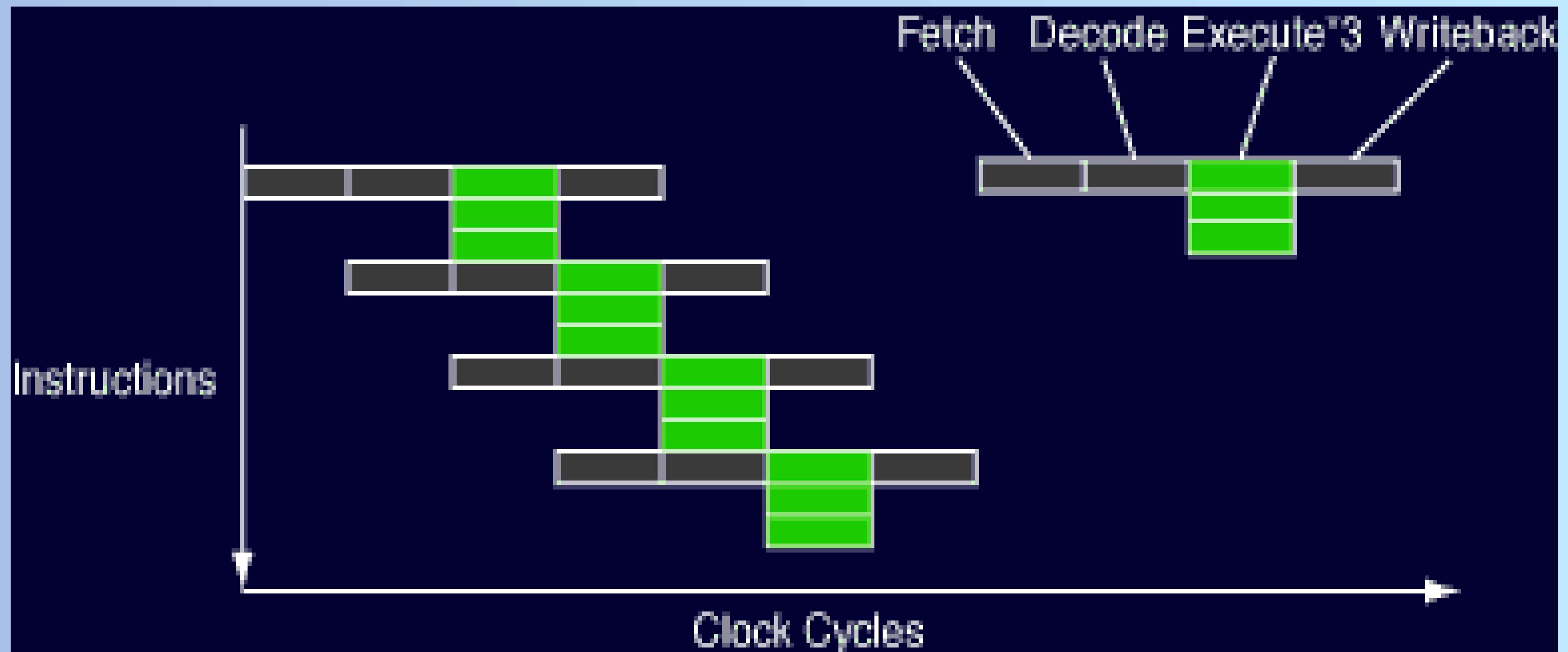
$$\Rightarrow \text{CPI} = 0.33 \quad (\text{IPC} = 3)$$

Kontrollfluss im superskalaren Prozessor (Superpipeline)



$$\Rightarrow \text{CPI} = 0.33 \text{ (IPC} = 3\text{)}$$

Kontrollfluss im VLIW – Prozessor



⇒ $CPI = 0.33$ ($IPC = 3$)

Minimierung von Pipeline – Blockierungen

RAW Datenkonflikt

- Forwarding
- Bypassing

WAW Datenkonflikt

- eine einzige Pipeline – Stufe zum Schreiben der Ergebnisse

WAR Datenkonflikt

- frühes Laden der Operanden in der Befehlsdekodierphase

Daten- und Steuerungskonflikte

- dynamische Umbenennung von Registern
- Änderungen der Befehlsreihenfolge unabhängiger Befehle durch Hardware oder Compiler
- Sprungzielvorhersage
- Spekulative Ausführung von Befehlen

strukturelle Konflikte

- Erweiterung knapper Ressourcen wie Register- und Speicherschnittstellen

2. ILP – Architekturen

- Superskalar – Technik
- VLIW – Technik
- EPIC – Technik
- Techniken im Vergleich

- mehr als ein Befehl pro Takt (superskalar)
- Zuweisung in Hardware (dynamisches Scheduling)
- Anzahl der zugewiesenen Befehle pro Takt
 - dynamisch von Hardware bestimmt
 - begrenzt durch maximal mögliche Zuweisungsbandbreite
- Komplexität des Hardware – Schedulers steigt mit
 - Größe des Befehlsfensters
 - Anzahl der Befehle, die außerhalb der Programmreihenfolge zugewiesen werden können
- Anzahl der Ausführungseinheiten
 - entspricht mindestens der Zuweisungsbandbreite
 - häufig größer um Strukturkonflikte zu vermeiden

- feste Anzahl von Befehlen wird vom Compiler
 - zu einem Befehlspaket zusammengepackt
 - in einem Maschinenwort meist fester Länge gespeichert
- alle Befehle eines Befehlspaketes müssen
 - unabhängig voneinander sein
 - eigene Opcodes und Operandenbezeichner enthalten
- Anzahl an Befehlen entspricht Anzahl an Ausführungseinheiten
- Befehlsanordnung
 - statisch durch den Compiler
 - von den Funktionseinheiten nicht änderbar
- Speicherhierarchie aus Cache und Hauptspeicher erschwert

- Weiterentwicklung der VLIW – Technik zur IA64 – Architektur
- erweitertes Dreibefehlsformat (dreifaches VLIW – Format)
- Ziel der EPIC – Technik:
 - Einfachheit und hohe Taktrate eines VLIW – Prozessors mit den Vorteilen des dynamischen Scheduling verbinden
- Compiler teilt dem Prozessor die Befehlsparallelität direkt mit
- EPIC – Prozessor muss im Idealfall
 - keine Überprüfung von Daten- und Kontrollflussabhängigkeiten durchführen
 - keine Veränderung der Ausführungsreihenfolge unterstützen
- Mikroarchitektur wird stark vereinfacht

EPIC – Befehlsbündel der IA64 – Architektur

- 128 Bit – Befehlsbündel
- drei IA64 – Befehle (41 Bit) und ein Template (5 Bit)
- Parallelität durch Template – Bits angegeben
 ⇒ keine Leerbefehle
- Befehle können daten – und kontrollflussabhängig sein
 ⇒ flexibler als VLIW – Formate



Issue Combinations for 2 Bundles (6 instructions)

second instruction group to issue

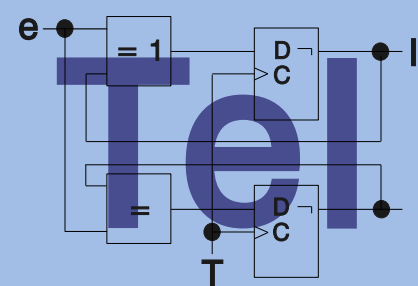
first instruction group to issue

	MII	MLX	MMI	MFI	MMF	MIB	MBB	BBB	MMB	MFB
MII	Green	Grey	Green	Green	Green	Green	Cyan	Cyan	Green	Cyan
MLX	Grey	Cyan	Green	Cyan	Green	Cyan	Cyan	Cyan	Green	Cyan
MMI	Green	Green	Green	Green	Green	Green	Green	Cyan	Green	Green
MFI	Green	Cyan	Green	Cyan	Green	Cyan	Cyan	Cyan	Green	Cyan
MMF	Green	Green	Green	Green	Green	Green	Green	Cyan	Green	Green
MIB	Green	Cyan	Green	Cyan	Green	Cyan	Cyan	Grey	Green	Cyan
MBB	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey
BBB	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey	Grey
MMB	Green	Green	Green	Green	Green	Green	Green	Grey	Green	Green
MFB	Cyan	Cyan	Green	Cyan	Green	Cyan	Cyan	Grey	Green	Cyan

Only the Itanium 2 processor can fully issue these combinations

Both Itanium and Itanium 2 processors can issue these

Not fully issued by either processor



Phasen der parallelen Ausführung

- (1) Abhängigkeitsanalyse und Gruppierung parallel ausführbarer Instruktionen
- (2) Zuweisung der Instruktion zu den Funktionseinheiten der Hardware
- (3) Start der Ausführung der Instruktionen

	Gruppierung	Zuweisung	Initiierung
Superskalar	Hardware	Hardware	Hardware
EPIC	Compiler	Hardware	Hardware
Dynamic VLIW	Compiler	Compiler	Hardware
VLIW	Compiler	Compiler	Compiler

Beispiel

- Architektur mit Funktionseinheiten: LSU 0, LSU 1, IU, BU
- zugehörige Latenzzeiten: 2, 2, 1, 2
- einfache Addition: $C = A + B$

Superskalar

Load R1, A
 Load R2, B
 Add R3, R1, R2
 Store C, R3

	LSU 0	VLIW LSU 1	IU	BU
Load R1, A	Load R1, A	Load R2, B	NOP	NOP
Load R2, B	NOP	NOP	NOP	NOP
Add R3, R1, R2	NOP	NOP	Add R3, R1, R2	NOP
Store C, R3	Store C, R3	NOP	NOP	NOP

Beispiel

- Architektur mit Funktionseinheiten: LSU 0, LSU 1, IU, BU
- zugehörige Latenzzeiten: 2, 2, 1, 2
- einfache Addition: $C = A + B$

EPIC

[2] Load R1, A
 [1] Load R2, B
 [1] Add R3, R1, R2
 [0] Store C, R3

Dynamic VLIW

LSU 0	LSU 1	IU	BU
Load R1, A	Load R2, B	NOP	NOP
NOP	NOP	Add R3, R1, R2	NOP
Store C, R3	NOP	NOP	NOP

- **Architektur- vs. Mikroarchitekturtechnik**
 - VLIW / EPIC Architekturtechniken
 - Superskalar Mikroarchitekturtechnik
- **Befehlsablaufplanung und Konfliktvermeidung**
 - VLIW / EPIC durch Compiler (statisch)
 - Superskalar durch Hardware (dynamisch)
- **Compileroptimierungen**
 - optimierender Compiler erforderlich
 - gleiche Optimierungsstrategien bei der Code – Generierung
 - VLIW / EPIC Zeitbedarf der Speicherzugriffe

- **Befehlsanordnung**

- Superskalar Befehlsstrom einfacher Befehle
- VLIW Befehlsstrom von VLIW – Befehlspaketen
- EPIC Befehlsstrom von EPIC – Befehlsbündeln

- **Reaktion auf Laufzeitergebnisse**

- abnehmende Flexibilität: Superskalar → EPIC → VLIW

- **Speicherorganisation**

- Superskalar / EPIC Verwendung von Speicher – Hierarchie
- VLIW Speicher – Hierarchie beeinflusst Latenzen

- **Sprungvorhersage und Sprungspekulation**

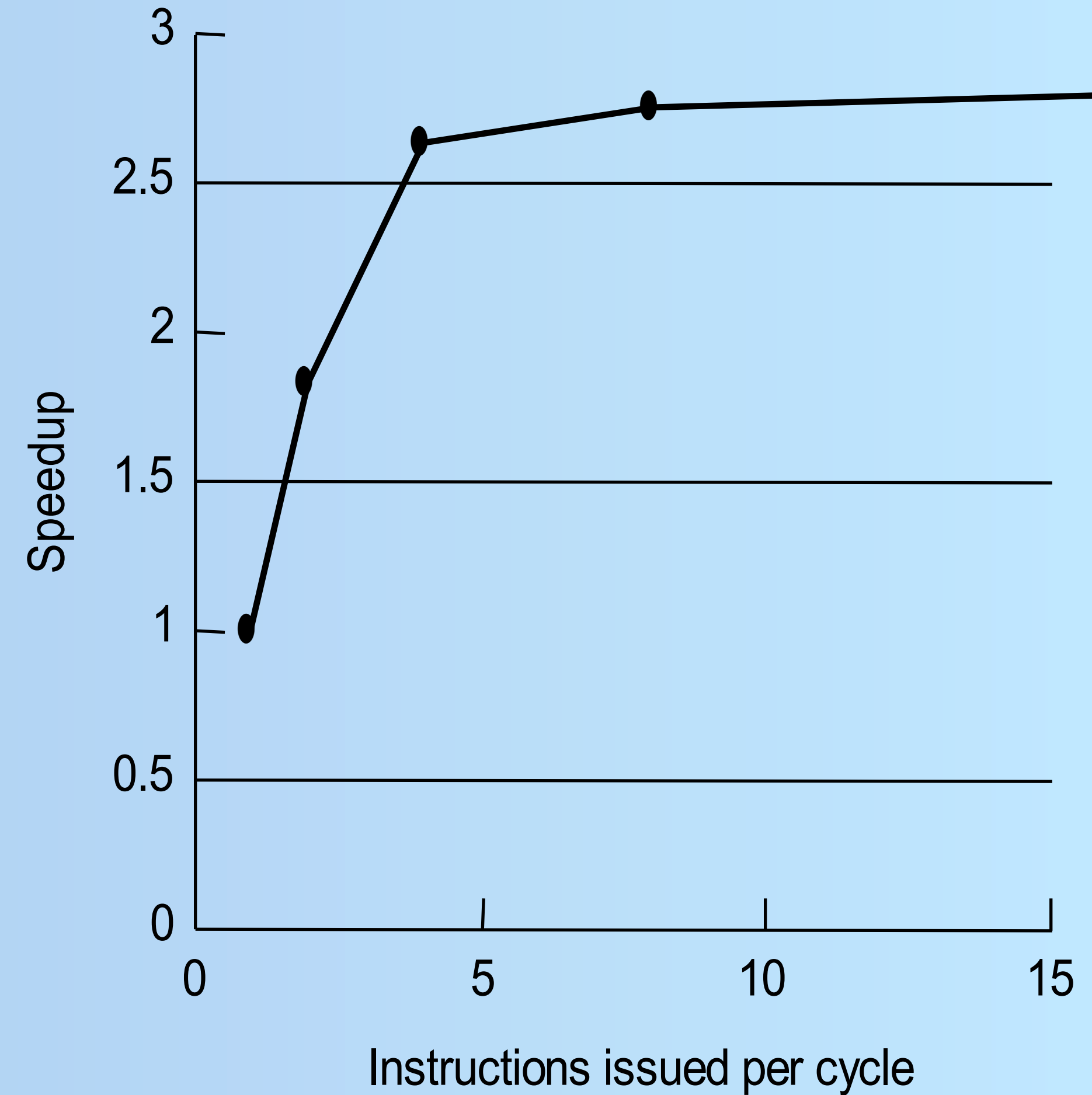
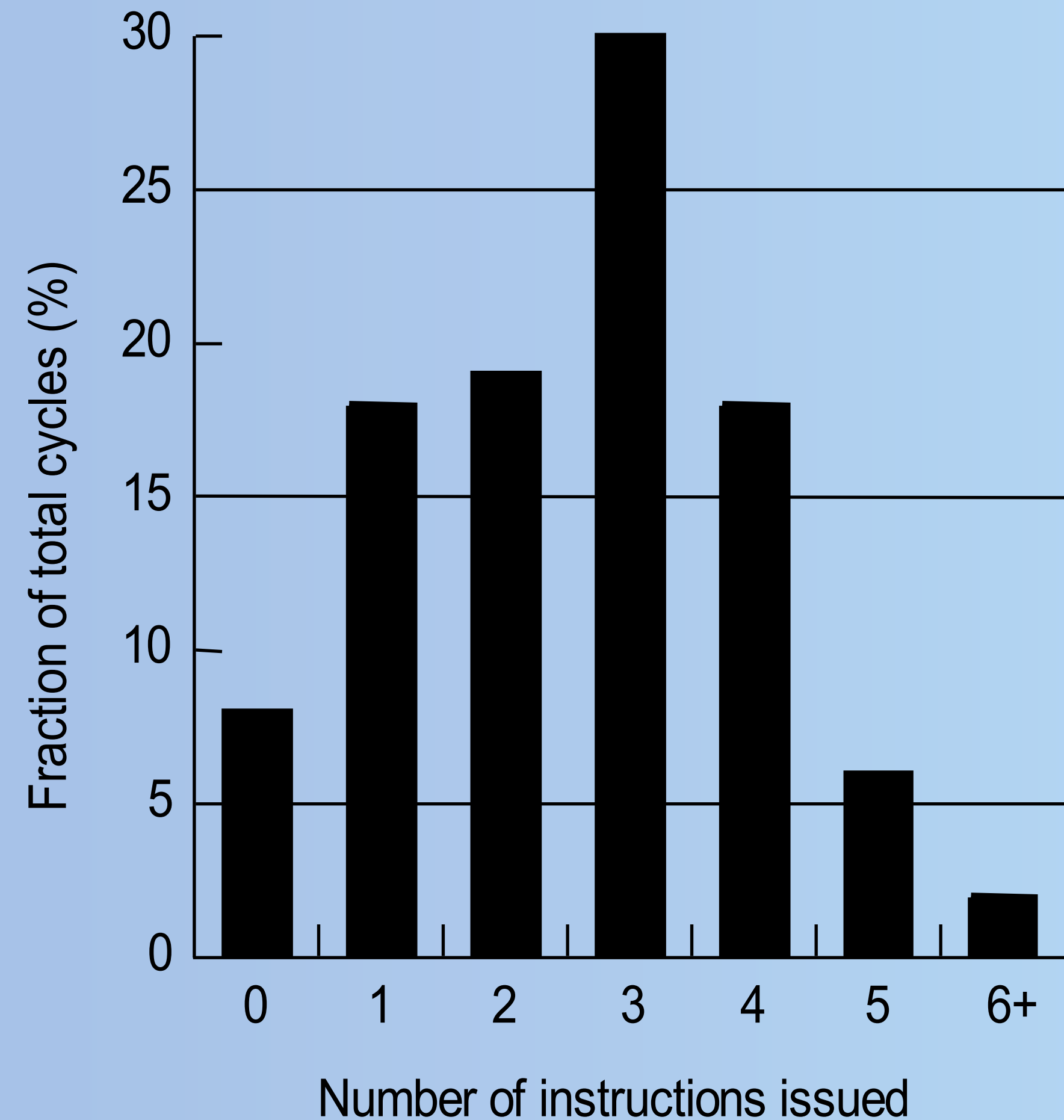
- Superskalar Standard bei heutigen Prozessoren
- VLIW / EPIC nur erschwert möglich

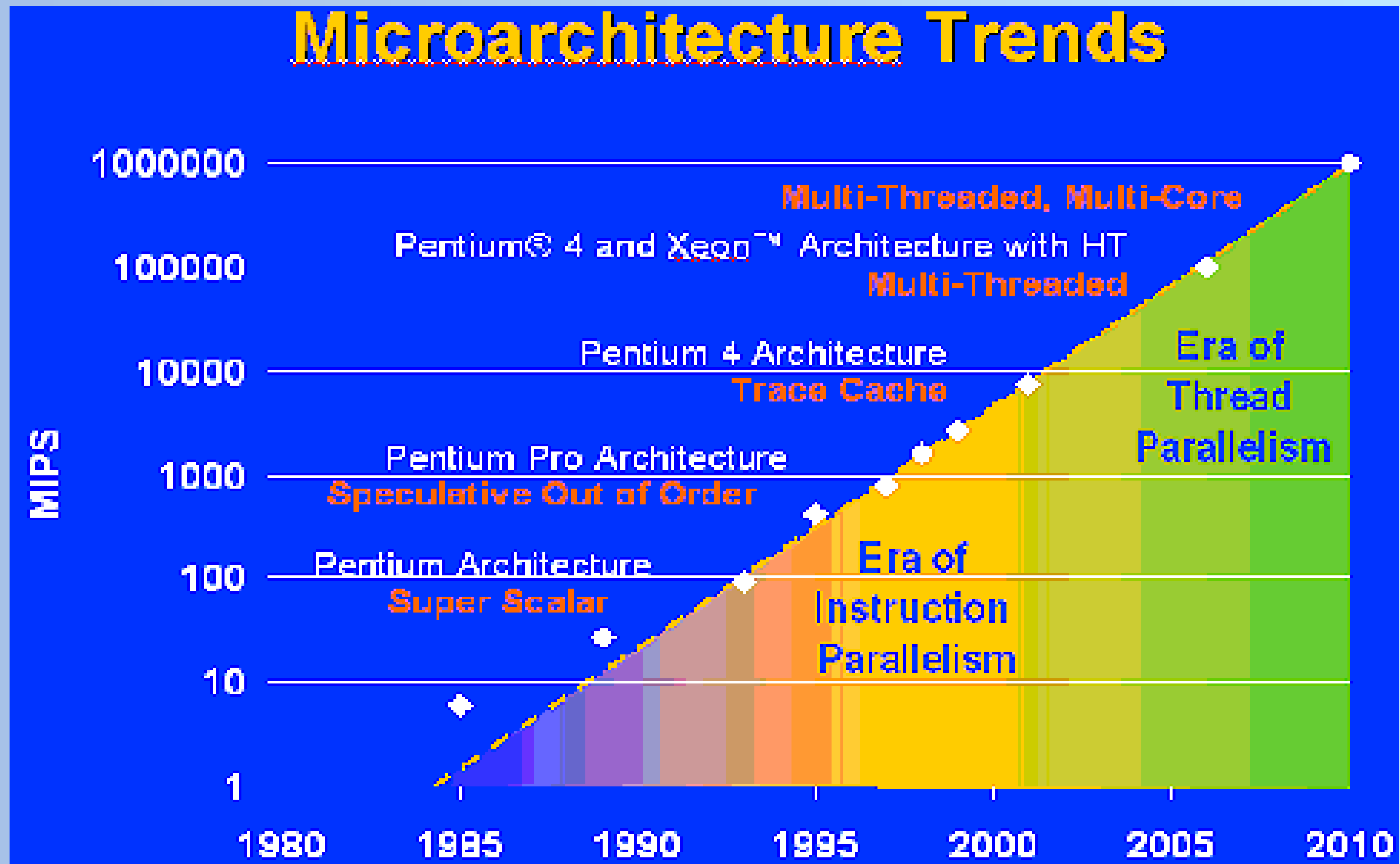
- **Codedichte**
 - VLIW festes Befehlsformat
abhängig vom Grad der zur Verfügung stehenden
Befehlsebenenparallelität
 - EPIC Redundanz durch Template – Bits
- **Erzielbare Verarbeitungsleistung und Anwendungsfehler**
 - VLIW Einfachheit der Prozessoren \Rightarrow höhere Taktrate
bei Code mit sehr hohem Parallelitätsgrad vorteilhaft
 - EPIC Starrheit des VLIW – Prinzips vermieden
Komplexität der Zuordnungseinheit durch das VLIW –
artige EPIC – Format verringert

3. Ausblick

- Zukunft von ILP
- Zukunft von EPIC

Parallelität in Programmen





- **geringfügige** Erhöhung von ILP
 - ⇒ **exponentielles** Wachstum der Hardware – Komplexität und Leistungsaufnahme
- **geringfügige** Erhöhung von TLP
 - ⇒ **exponentielles** Wachstum der Programmier – und Debugging – Zeit
- single thread performance zur Abgrenzung von Multicore – Architekturen

- Mark Smotherman: Understanding EPIC Architectures and Implementations (http://www.cs.clemson.edu/~mark/464/acmse_epic.pdf)
- Michael S. Schlansker, B. Ramakrishna Rau: EPIC: An Architecture for Instruction-Level Parallel Processors (<http://www.hpl.hp.com/techreports/1999/HPL-1999-111.pdf>)
- Johan De Gelas: Itanium - is there light at the end of the tunnel? (<http://www.anandtech.com/cpuchipsets/showdoc.aspx?i=2598>)
- Jason Patterson: Modern Microprocessors (<http://www.pattosoft.com.au/Articles/ModernMicroprocessors/>)
- Michael S. Schlansker, B. Ramakrishna Rau: Achieving High Levels of Instruction-Level Parallelism with Reduced Hardware Complexity(<http://www.hpl.hp.com/techreports/96/HPL-96-120.pdf>)
- Wikipedia:
 - http://en.wikipedia.org/wiki/Very_long_instruction_word
 - http://en.wikipedia.org/wiki/Explicitly_parallel_instruction_computing