Rechnerstrukturen

Vorlesung im Sommersemester 2009

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

Institut für Technische Informatik

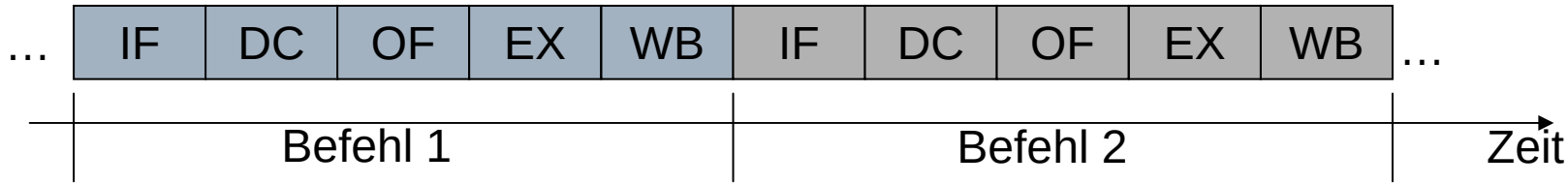
Kapitel 1: Grundlagen

1.5 Parallelverarbeitung

- Sequentielle Sichtweise
  - Berechnungen werden schrittweise oder seriell ausgeführt
  - Algorithmen sind als Folge von Berechnungsschritten organisiert
  - Sequentielle Programme
  - Schrittweise Ausführung einer Folge von Befehlen auf einer sequentiellen Maschine
- Beschleunigung der Ausführung
  - Erhöhung der Taktfrequenz
  - Parallele Ausführung der Aufgaben

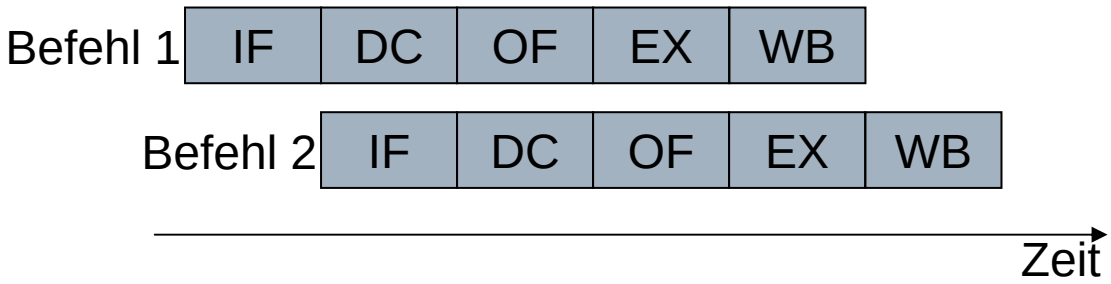
- Parallelverarbeitung
  - Algorithmenentwurf
    - Muss viele unabhängige Operationen umfassen
  - Struktur der Programmiersprache
    - Muss die Spezifikation oder die automatische Identifizierung paralleler Operationen ermöglichen
  - Rechnerarchitektur
    - Gleichzeitige Ausführung paralleler Operationen

- Sequentieller Rechner
  - Sequentielle Ausführung einer Folge von Maschinenbefehlen
    - Maschinenbefehlszyklus

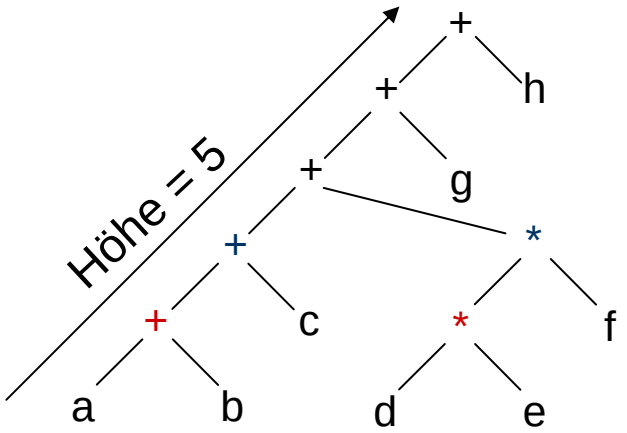


IF: Befehl holen
DK: Befehl dekodieren
OF: Operanden bereitstellen
EX: Befehl ausführen
WB: Ergebnis speichern

- Pipelining des Maschinenbefehlszyklus
  - Überlappte Ausführung der Phasen des Maschinenbefehlszyklus



- Gleichzeitige Ausführung von Operationen
  - Beispiel: Auswertung des arithmetischen Ausdrucks *EXP= a+b+c+(d*e*f)+g+h* durch den Compiler von links nach rechts



Seq. Ausführung
S1: t1 = a + c
S2: t1 = t1 + c
S3: t2 = d * e
S4: t2 = t2 * f
S5: t1 = t1 + t2
S5: t1 = t1 + g
S6: t1 = t1 + h

Parallele Ausf.
S1: t1 = a + c; S3: t2 = d * e
S2: t1 = t1 + c; S4: t2 = t2 * f
S5: t1 = t1 + t2
S5: t1 = t1 + g
S6: t1 = t1 + h

  - Höhe:
    - längster Pfad von der Wurzel zum Blatt
    - Minimale Anzahl von Schritten zur Auswertung des Ausdrucks

- Formen des Parallelismus
  - Nebenläufigkeit
    - Eine Maschine arbeitet nebenläufig, wenn die Objekte vollständig gleichzeitig abgearbeitet werden.
  - Pipelining
    - Pipelining auf einer Maschine liegt dann vor, wenn die Bearbeitung eines Objektes in Teilschritte zerlegt und diese in einer sequentiellen Folge (Phasen der Pipeline) ausgeführt werden. Die Phasen der Pipeline können für verschiedene Objekte überlappt abgearbeitet werden. (Bode 95)

- Ebenen der Parallelität
  - Programmebene
    - Parallele Verarbeitung verschiedener Programme
    - Vollständig unabhängige Einheiten
      - ohne gemeinsame Daten
      - wenig oder keine Kommunikation und Synchronisation
    - Parallelverarbeitung wird vom Betriebssystem organisiert

- Ebenen der Parallelität
  - Prozessebene (Task-Ebene)
    - Programm wird in Anzahl parallel ausführbarer Prozesse zerlegt
      - Prozess: schwergewichtiger Prozess (heavy-weighted process), Beispiel: UNIX-Prozesse
        - » Besteht aus vielen sequentiell ausgeführten Befehlen und umfasst eigene Datenbereiche
      - Synchronisation und Kommunikationsaufwand
    - Betriebssysteme unterstützt Parallelverarbeitung durch Primitive zur Prozessverwaltung, Prozess-Synchronisation, Prozesskommunikation

- Ebenen der Parallelität
  - Blockebene
    - leichtgewichtige Prozesse (Threads)
      - Bestehen jeweils aus sequentiell ausgeführten Befehlen teilen sich gemeinsamen Adressraum
      - Beispiel: Threads gemäß POSIX 1003.a Standard in mehrfädigen (multithreaded) Betriebssystemen
      - Synchronisation über Schlossvariablen (mutex), und Bedingungsvariablen (condition variables) oder darauf aufbauenden Synchronisationsmechanismen
      - Kommunikation über gemeinsame Daten
      - Aufwand für Thread-Erzeugung und-Beendigung, Thread-Wechsel geringer

- Ebenen der Parallelität
  - Blockebene
    - Anweisungsblöcke
      - Innere und äußere parallele Schleifen in FORTRAN-Dialekten
      - Verwendung von Microtasking
      - Hohes Parallelitätspotential durch parallel ausführbare Schleifeninterationen

- Ebenen der Parallelität
  - Anweisungs- oder Befehlsebene
    - Parallele Ausführung einzelner Maschinenbefehle oder elementarer Anweisungen
    - Optimierende (parallelisierende) Compiler für VLIW-Prozessoren oder Anwendung der Superskalartechnik in superskalaren Mikroprozessoren
      - Analyse der sequentiellen Befehlsfolge
      - Umordnen und Parallelisieren der Befehle
    - Datenflusssprachen und funktionale Programmiersprachen erlauben explizite Spezifikation der Parallelität

- Ebenen der Parallelität
  - Suboperationsebene
    - Elementare Anweisung wird durch Compiler oder durch die Maschine in Suboperationen aufgebrochen, die parallel ausgeführt werden
      - Vektoroperationen
        - » Überlappte Ausführung auf Vektorrechner in Vektorpipeline
        - » Komplexe Datenstrukturen (Matrizen, Vektoren, Datenströme) sind in höherer Programmiersprache verfügbar oder werden von einem parallelisierenden, vektorisierenden Compiler aus einer sequentiellen Programmiersprache generiert
        - » Beispiel: Vektoraddition C = A + B statt Abarbeitung einer Schleife

- Körnigkeit der Parallelität
  - Die Körnigkeit oder Granularität (grain size) ergibt sich aus dem Verhältnis von Rechenaufwand zu Kommunikations- oder Synchronisationsaufwand. Sie bemisst sich nach der Anzahl der Befehle in einer sequentiellen Befehlsfolge.
  - Programm-, Prozess- und Blockebene werden häufig auch als grobkörnige (large grained) Parallelität,
  - die Anweisungsebene als feinkörnige (finely grained) Parallelität bezeichnet.
  - Seltener wird auch von mittelkörniger (medium grained) Parallelität gesprochen, dann ist meist die Blockebene gemeint.

- # Techniken der Parallelarbeit vs. Parallelitätsebenen

Parallelarbeitstechniken	Programmebene	Prozessebene	Blockebene	Anweisungsebene	Suboperationsebene
Techniken der Parallelarbeit durch Rechnerkopplung					
Grid-Computing	X	X			
Cluster	X	X			
Techniken der Parallelarbeit durch Prozessorkopplung					
Nachrichtenkopplung	X	X			
Speicherkopplung (SMP)	X	X	X		
Speicherkopplung (DSM)	X	X	X		
Grobkörniges Datenflußprinzip		X	X		
Techniken der Parallelarbeit in der Prozessorarchitektur					
Befehlspipelining				X	
Superskalar				X	
VLIW				X	
Überlappung von E/A- mit CPU-Operationen				X	
Feinkörniges Datenflußprinzip				X	
SIMD-Techniken					
Vektorrechnerprinzip					X
Feldrechnerprinzip					X
SIMD-Operationen					

Quelle: Ungerer, T. : Skript Rechnerstrukturen, SS 2000

Einführung: Literatur

- Theo Ungerer: Parallelrechner und parallele Programmierung, Kap.1.1 – 1.3

Kapitel 1: Grundlagen

1.5 Einführung: Klassifizierung von Rechnerarchitekturen

- Klassifikationen
  - Aufspannen von Entwurfsräumen
  - Aufzeigen von Entwurfsalternativen
  - Klassifikationsschemata versuchen, der Vielfalt von Rechnerarchitekturen eine Ordnungsstruktur zu geben
  - Frühe Klassifikationen konzentrieren sich auf die Hardware-Struktur
    - Anordnung und Organisation der Verarbeitungselemente
    - Operationsprinzip

- Klassifizierung nach M. Flynn
  - Zweidimensionale Klassifizierung
    - Hauptkriterien:
      - Zahl der Befehlsströme und
      - Zahl der Datenströme sind.
    - Merkmale:
      - Ein Rechner bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Befehl.
      - Ein Rechner bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Datenwert.

- Klassifizierung nach M. Flynn
  - Vier Klassen von Rechnerarchitekturen
    - SISD Single Instruction – Single Data
      - Uniprozessor
    - SIMD Single Instruction – Multiple Data
      - Vektorrechner, Feldrechner
    - MISD Multiple Instructions – Single Data
      - ?
    - MIMD Multiple Instructions – Multiple Data
      - Multiprozessor

- Diskussion der Flynnschen Klassifizierung
  - Schwachpunkte:
    - Sehr hohes Abstraktionsniveau ==> sehr unterschiedliche Rechnerarchitekturen fallen in die gleiche Klasse.
    - Viele moderne Parallelarbeitstechniken (z.B. Superskalar, Befehlspipelining, VLIW) lassen sich überhaupt nicht einordnen.
    - In heutigen Rechnern findet man die Kombination mehrerer Techniken.
      Beispiel: Vektorrechner-Multiprozessoren fallen in eine als MIMD/SIMD zu bezeichnende Klasse.
    - Die Klasse MISD wurde nur der Systematik wegen aufgeführt.

- Diskussion von Klassifikationen
  - Kennzeichnend für moderne Rechnerstrukturen:
    - Prinzip der Virtualität:
      - Mit verschiedenen Techniken und Mechanismen in der Hardware können auf einer Maschine verschiedene parallele Programmiermodelle unterstützt werden
      - Die zugrunde liegende Organisation und Architektur ist weitgehend transparent
    - Allgemeiner Trend in der Rechnerarchitektur
      - Verwendung von Standardkomponenten
      - Verständnis über die Implementierungstechniken zur Virtualität
      - Keine allgemeingültige Klassifikation!

Kapitel 1: Grundlagen

1.5 Einführung: Klassifizierung von Rechnerarchitekturen

- Klassifikationen
  - Aufspannen von Entwurfsräumen
  - Aufzeigen von Entwurfsalternativen
  - Klassifikationsschemata versuchen, der Vielfalt von Rechnerarchitekturen eine Ordnungsstruktur zu geben
  - Frühe Klassifikationen konzentrieren sich auf die Hardware-Struktur
    - Anordnung und Organisation der Verarbeitungselemente
    - Operationsprinzip

- Klassifizierung nach M. Flynn
  - Zweidimensionale Klassifizierung
    - Hauptkriterien:
      - Zahl der Befehlsströme und
      - Zahl der Datenströme sind.
    - Merkmale:
      - Ein Rechner bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Befehl.
      - Ein Rechner bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Datenwert.

- Klassifizierung nach M. Flynn
  - Vier Klassen von Rechnerarchitekturen
    - SISD Single Instruction – Single Data
      - Uniprozessor
    - SIMD Single Instruction – Multiple Data
      - Vektorrechner, Feldrechner
    - MISD Multiple Instructions – Single Data
      - ?
    - MIMD Multiple Instructions – Multiple Data
      - Multiprozessor

- Diskussion der Flynnschen Klassifizierung
  - Schwachpunkte:
    - Sehr hohes Abstraktionsniveau ==> sehr unterschiedliche Rechnerarchitekturen fallen in die gleiche Klasse.
    - Viele moderne Parallelarbeitstechniken (z.B. Superskalar, Befehlspipelining, VLIW) lassen sich überhaupt nicht einordnen.
    - In heutigen Rechnern findet man die Kombination mehrerer Techniken.
      Beispiel: Vektorrechner-Multiprozessoren fallen in eine als MIMD/SIMD zu bezeichnende Klasse.
    - Die Klasse MISD wurde nur der Systematik wegen aufgeführt.

- Diskussion von Klassifikationen
  - Kennzeichnend für moderne Rechnerstrukturen:
    - Prinzip der Virtualität:
      - Mit verschiedenen Techniken und Mechanismen in der Hardware können auf einer Maschine verschiedene parallele Programmiermodelle unterstützt werden
      - Die zugrunde liegende Organisation und Architektur ist weitgehend transparent
    - Allgemeiner Trend in der Rechnerarchitektur
      - Verwendung von Standardkomponenten
      - Verständnis über die Implementierungstechniken zur Virtualität
      - Keine allgemeingültige Klassifikation!

Kapitel 1: Grundlagen

1.5 Einführung: Klassifizierung von Rechnerarchitekturen

- Klassifikationen
  - Aufspannen von Entwurfsräumen
  - Aufzeigen von Entwurfsalternativen
  - Klassifikationsschemata versuchen, der Vielfalt von Rechnerarchitekturen eine Ordnungsstruktur zu geben
  - Frühe Klassifikationen konzentrieren sich auf die Hardware-Struktur
    - Anordnung und Organisation der Verarbeitungselemente
    - Operationsprinzip

- Klassifizierung nach M. Flynn
  - Zweidimensionale Klassifizierung
    - Hauptkriterien:
      - Zahl der Befehlsströme und
      - Zahl der Datenströme sind.
    - Merkmale:
      - Ein Rechner bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Befehl.
      - Ein Rechner bearbeitet zu einem gegebenen Zeitpunkt einen oder mehr als einen Datenwert.

- Klassifizierung nach M. Flynn
  - Vier Klassen von Rechnerarchitekturen
    - SISD Single Instruction – Single Data
      - Uniprozessor
    - SIMD Single Instruction – Multiple Data
      - Vektorrechner, Feldrechner
    - MISD Multiple Instructions – Single Data
      - ?
    - MIMD Multiple Instructions – Multiple Data
      - Multiprozessor

- Diskussion der Flynnschen Klassifizierung
  - Schwachpunkte:
    - Sehr hohes Abstraktionsniveau ==> sehr unterschiedliche Rechnerarchitekturen fallen in die gleiche Klasse.
    - Viele moderne Parallelarbeitstechniken (z.B. Superskalar, Befehlspipelining, VLIW) lassen sich überhaupt nicht einordnen.
    - In heutigen Rechnern findet man die Kombination mehrerer Techniken.
      Beispiel: Vektorrechner-Multiprozessoren fallen in eine als MIMD/SIMD zu bezeichnende Klasse.
    - Die Klasse MISD wurde nur der Systematik wegen aufgeführt.

- Diskussion von Klassifikationen
  - Kennzeichnend für moderne Rechnerstrukturen:
    - Prinzip der Virtualität:
      - Mit verschiedenen Techniken und Mechanismen in der Hardware können auf einer Maschine verschiedene parallele Programmiermodelle unterstützt werden
      - Die zugrunde liegende Organisation und Architektur ist weitgehend transparent
    - Allgemeiner Trend in der Rechnerarchitektur
      - Verwendung von Standardkomponenten
      - Verständnis über die Implementierungstechniken zur Virtualität
      - Keine allgemeingültige Klassifikation!

- **Kapitel 2: Parallelismus auf Befehlsebene**

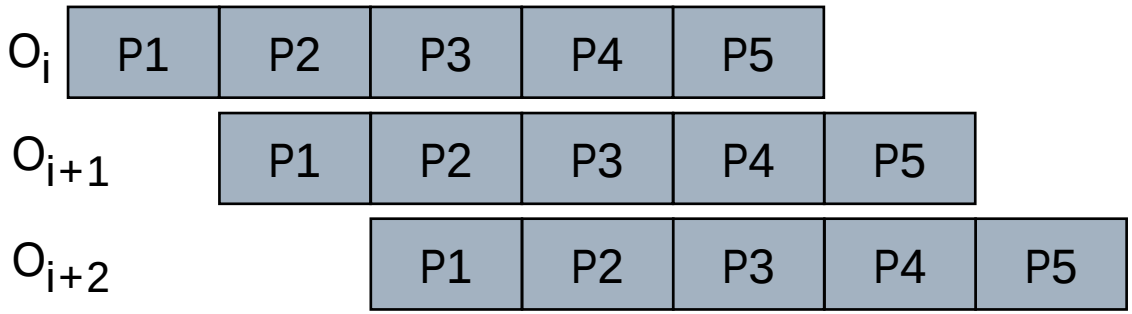
2.1: Pipelining

Parallelismus auf Befehlsebene

- Überblick
  - Pipelining
    - Überlappte Ausführung der Phasen des Maschinenbefehlszyklus
      - Nützen alle Prozessoren seit 1985 aus
  - Nebenläufigkeit
    - Zu einem Zeitpunkt gleichzeitige Ausführung mehrerer Maschinenbefehle zu
      - Dynamische Ansätze
        - » Superskalare Mikroprozessoren
      - Statische Ansätze
        - » VLIW, EPIC

Parallelismus auf Befehlsebene

- Pipelining
  - *Pipelining auf einer Maschine liegt dann vor, wenn die Bearbeitung eines Objektes in Teilschritte zerlegt und diese in einer sequentiellen Folge (Phasen der Pipeline) ausgeführt werden. Die Phasen der Pipeline können für verschiedene Objekte überlappt abgearbeitet werden. (Bode 95)*



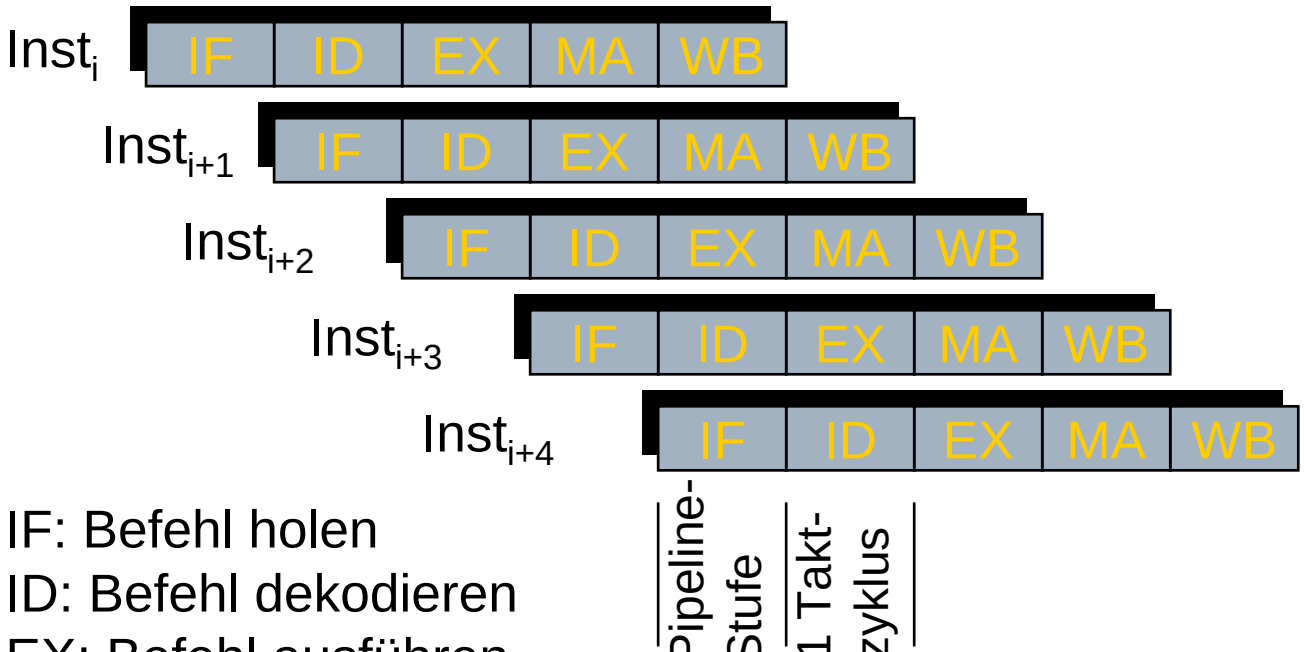
Parallelismus auf Befehlsebene

- Pipelining
  - Befehlspipelining (Instruction Pipelining):
    - Zerlegung der Ausführung einer Maschinenoperation in Teilphasen, die dann von hintereinander geschalteten Verarbeitungseinheiten taktsynchron bearbeitet werden, wobei jede Einheit genau eine spezielle Teiloperation ausführt.
  - Pipeline:
    - Gesamtheit der Verarbeitungseinheiten
  - Pipeline-Stufe:
    - Stufen der Pipeline, die jeweils durch Pipeline-Register getrennt sind

Parallelismus auf Befehlsebene

- RISC (Reduced Instruction Set Computers)
  - Einfache Maschinenbefehle
    - Einheitliches und festes Befehlsformat
  - Load/Store Architektur
    - Befehle arbeiten auf Registeroperanden
    - Lade- und Speicherbefehle greifen auf Speicher zu
  - Einzyklus-Maschinenbefehle
    - Effizientes Pipelining des Maschinenbefehlszyklus
    - Einheitliches Zeitverhalten der Maschinenbefehle, wovon nur Lade- und Speicherbefehle sowie die Verzweigungsbefehle abweichen
  - Optimierende Compiler
    - Reduzierung der Befehle im Programm

- Implementierung eines RISC-Befehlssatzes
  - k-stufige Befehlspipeline (k=5)



IF: Befehl holen
ID: Befehl dekodieren
EX: Befehl ausführen
MA: Speicherzugriff
WB: Zurückschreiben

- Leistungsaspekte
  - Ausführungszeit eines Befehls:
    - Zeit, die zum Durchlaufen der Pipeline benötigt wird
    - Ausführung eines Befehls in k Taktzyklen (ideale Verhältnisse)
    - Gleichzeitige Behandlung von k Befehlen (ideale Verhältnisse)
  - Latenz:
    - Anzahl der Zyklen zwischen einer Operation, die ein Ergebnis produziert, und einer Operation, die das Ergebnis verwendet

Pipelining

- Leistungsaspekte
  - Laufzeit T

$$T = n+k-1$$

    - N: Anzahl der Befehle in einem Programm
    - Annahme: ideale Verhältnisse!
  - Beschleunigung S

$$S = n * k / (k + n - 1)$$