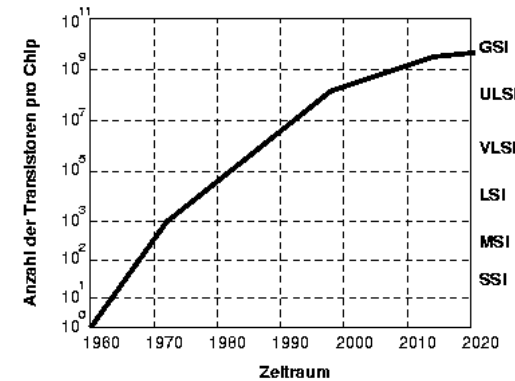


Kapitel 2

Pipeline-Verarbeitung



Technologie- Entwicklung



SSI: Small Scale Integration
MSI: Medium Scale Integration
LSI: Large Scale Integration
VLSI: Very Large Scale Integration
ULSI: Ultra Large Scale Integration
GSI: Giga Scale Integration



Leistungssteigerung in Rechnersystemen

Welche Möglichkeiten hat man prinzipiell zur Leistungssteigerung in Rechnersystemen?

➤ **Technologische Maßnahmen:**

Anwendung schnellerer Technologien,
Redesign ist nötig.

➤ **Strukturelle Maßnahmen:**

z. B. Zahl der Transistoren erhöhen
→ Parallelarbeit



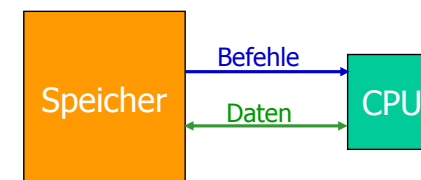
Strukturelle Maßnahmen

Klassifikation von Rechnerstrukturen nach Flynn:

Unterscheidung bezüglich der gleichzeitig bearbeiteten
Befehls- und Datenströme:

❑ **SISD (Single Instruction Single Data):**

Ein Datenstrom wird entsprechend einer seriellen
Befehlsfolge verarbeitet (von-Neumann-Rechner)



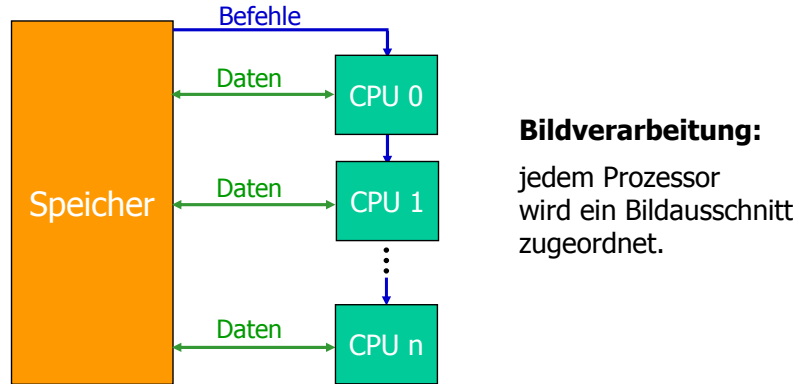
IBM-PC, IBM 370,
Micro-VAX von DEC



Strukturelle Maßnahmen

□ SIMD (Single Instruction Multiple Data):

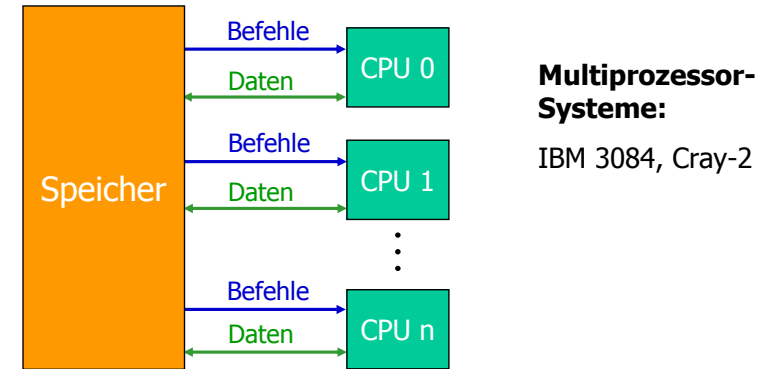
Alle Prozessoren führen gleichzeitig **dieselben Befehle** auf **verschiedenen Daten** aus (Array-Prozessoren).



Strukturelle Maßnahmen

□ MIMD (Multiple Instruction Multiple Data):

Alle Prozessoren führen gleichzeitig **verschiedene Befehle** auf **verschiedenen Daten** aus.



Strukturelle Maßnahmen

□ MISD (Multiple Instruction Single Data):

- Es wird nur ein Datenstrom bearbeitet.
- Bestimmte Ausführungseinheiten übernehmen die Ausführung bestimmter Teile einer Operation (Pipeline-Verarbeitung)
- Parallelität auf Befehlsebene.
- „Moderne“ Prozessoren: ab Intel 80286



Zusammenfassung: Strukturelle Maßnahmen

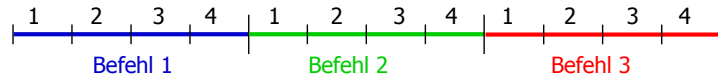
- **Mehrprozessorsysteme:** Mehrere Prozessoren mit unabhängigen Programmen arbeiten mit einem gemeinsamen Hauptspeicher
- **Feldrechner:** Mehrere Prozessoren arbeiten am gleichen Programm, aber mit verschiedenen Daten (Bsp: Bildverarbeitung)
- System mit **funktionsspezialisierten** Prozessoren: Mehrere Spezialprozessoren arbeiten unter einer CPU und mit einem Hauptspeicher
- **Fließbandverarbeitung (Pipeline-Struktur):** In einer Kette von Prozessoren übernimmt jeder die Ausführung bestimmter Teile einer Operation.



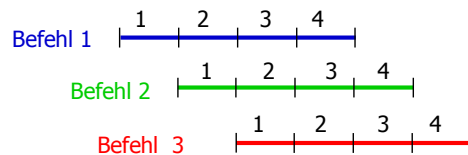
2. 1 Pipeline-Verarbeitung

Ausführung von 3 gleichartigen Verarbeitungsaufträgen
in 4 Teilverarbeitungsschritten:

Serielle Verarbeitung:



Pipeline-Verarbeitung:



Pipelining „Fließband-Bearbeitung“

„Pipelines beschleunigen
die Ausführungsgeschwindigkeit eines
Rechners in gleicher Weise wie Henry Ford
die Autoproduktion mit der Einführung des
Fließbandes revolutionierte.“

(Peter Wayner 1992)



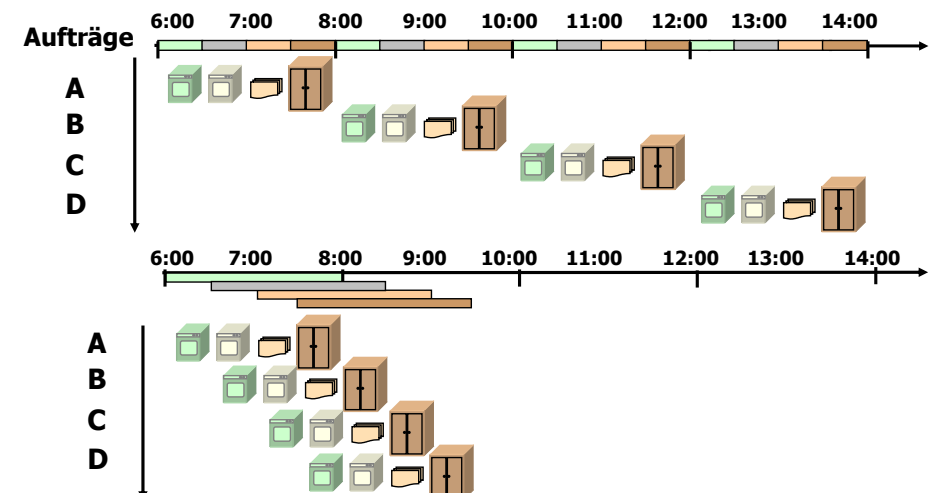
Wäsche Pipelining

Ein Wäsche-Vorgang kann in 4 Teilvorgänge unterteilt
werden:

- Schmutzige Wäsche in die Waschmaschine
- Nasse Wäsche in den Trockner
- Falten, Bügeln, ...
- Kleider in den Schrank



Wäsche Pipelining



Pipeline Verarbeitung

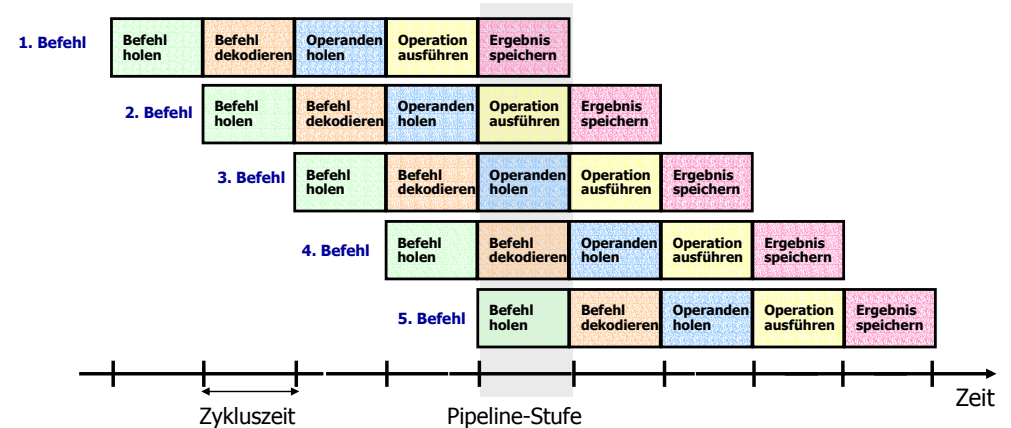
Oft ablaufende Operationen werden in eine Folge von Teilprozessen zerlegt. Für jeden Teilprozess wird ein spezieller Prozessor (spezielle Ausführungseinheit) vorgesehen.

Typischer Fall: Befehlsverarbeitung wird aufgeteilt in:

- Befehl holen
- Befehl decodieren (interpretieren)
- Operand(en) holen
- Operation ausführen
- Ergebnis speichern

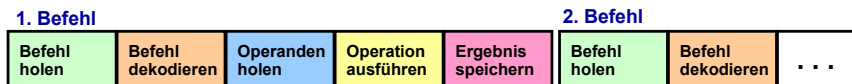


Einfache fünfstufige Befehlspipeline

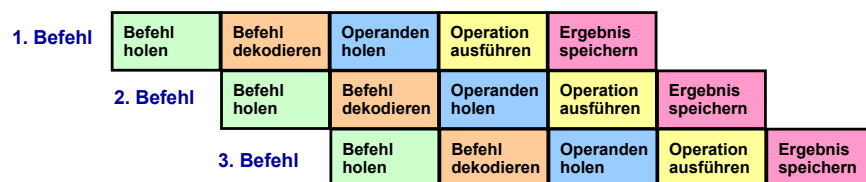


Pipelining

Sequentielle Ausführung:



Pipelining:

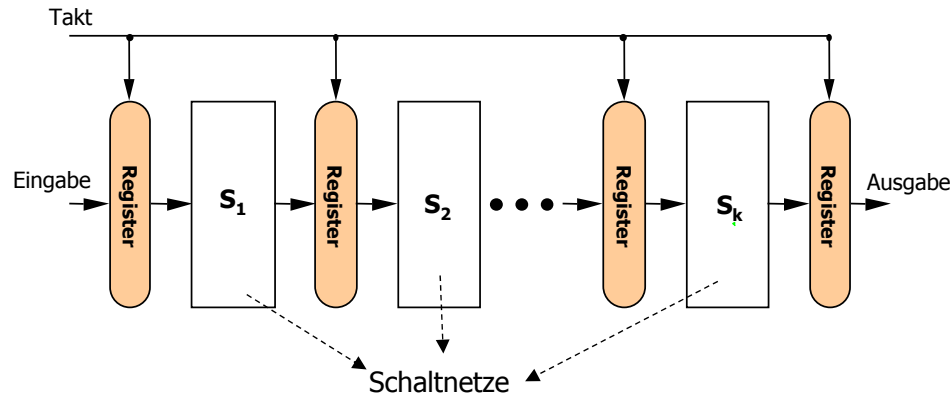


Definitionen

- ❑ **Pipelining:** Zerlegung einer Maschinenoperation in mehrere Phasen oder Suboperationen, die dann von hintereinander geschalteten Verarbeitungseinheiten taktsynchron bearbeitet werden, wobei jede Verarbeitungseinheit genau eine spezielle Teiloperation ausführt
- ❑ Die Gesamtheit dieser Verarbeitungseinheiten nennt man eine **Pipeline**.
- ❑ Bei einer **Befehlspipeline** (Instruction Pipeline) wird die Ausführung eines Maschinenbefehls in verschiedene Phasen unterteilt, aufeinanderfolgende Maschinenbefehle werden jeweils um einen Taktzyklus versetzt ausgeführt



2.2 Pipeline-Stufen und Pipeline-Register



Verzögerungszeiten:

- der Schaltnetze: τ_i ($i = 1, \dots, k$)
- der Pipeline-Register: τ_{reg}

Länge eines Taktzyklus:

$$\tau = \max\{\tau_1, \tau_2, \dots, \tau_k\} + \tau_{reg}$$



Definitionen

- Jede Stufe der Pipeline heißt **Pipeline-Stufe** oder **Pipeline-Segment**.
- Pipeline-Stufen werden durch getaktete **Pipeline-Register** (auch *latches* genannt) getrennt.
- Ein Pipeline-Maschinentakt ist die Zeit, die benötigt wird, um einen Befehl eine Stufe weiter durch die Pipeline zu schieben.
- Idealerweise wird ein Befehl in einer **k-stufigen Pipeline** in k Takten von k Stufen ausgeführt.
- Wird in jedem Takt ein neuer Befehl geladen, dann werden zu jedem Zeitpunkt unter idealen Bedingungen k Befehle gleichzeitig behandelt und jeder Befehl benötigt k Takte, bis zum Verlassen der Pipeline.



Definitionen

- ❑ **Latenz:** die Zeit, die ein Befehl benötigt, um alle k Pipeline-Stufen zu durchlaufen.

Ideale Verhältnisse:

- Ausführung eines Befehls in k Takten.
- Es werden gleichzeitig k Befehle bearbeitet.

- ❑ **Durchsatz einer Pipeline:** Anzahl der Befehle, die eine Pipeline pro Takt verlassen können. Dieser Wert spiegelt die Rechenleistung einer Pipeline wider.



Leistungssteigerung

n Befehle in einer Pipeline mit k Stufen

- Hypothetischen Prozessor ohne Pipeline:
 $n * k$ Taktzyklen
- Pipeline-Prozessor mit einer k -stufigen Pipeline:
 $k + (n-1)$ Taktzyklen (unter der Annahme idealer Bedingungen mit einer Latenz von k Takten und einem Durchsatz von 1)
 - k Taktzyklen, um die Pipeline zu füllen
 - $(n-1)$ Taktzyklen, um die restlichen $(n-1)$ Befehle auszuführen.

➔ Leistungssteigerung S (*speedup*):

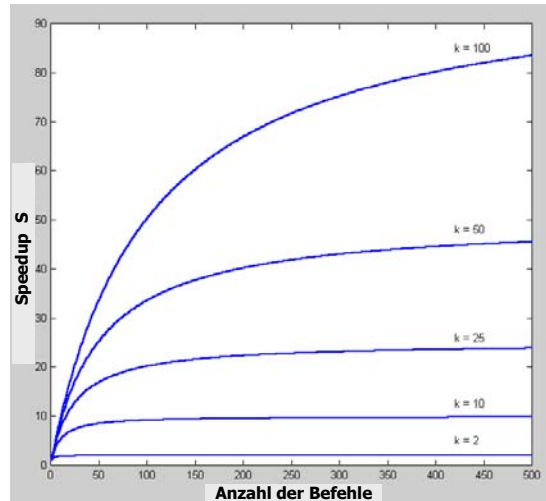
$$S = \frac{n * k}{k + (n-1)}$$



Leistungssteigerung

$$S = \frac{n * k}{k + (n-1)}$$

$$\lim_{n \rightarrow \infty} S = k$$



Pause



Befehl bereitstellen



Befehl dekodieren



Operanden holen



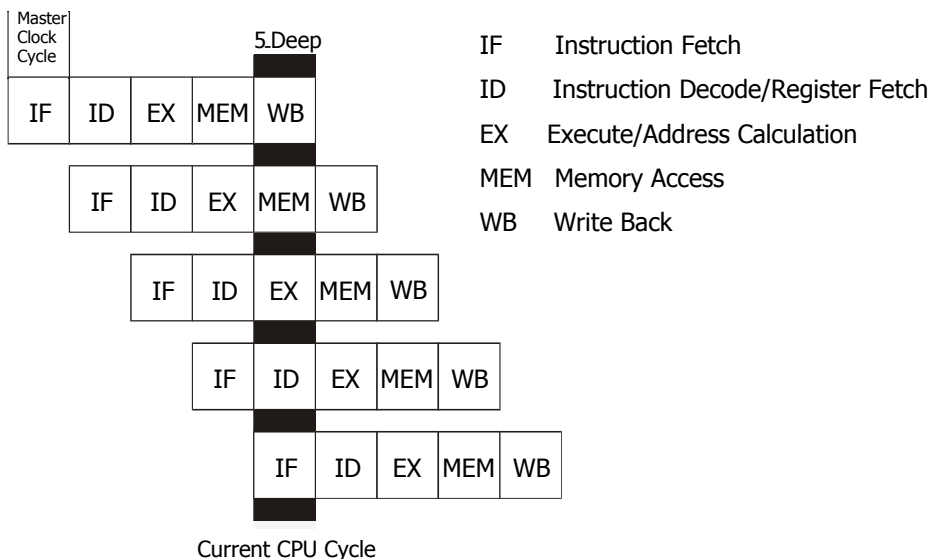
Operation ausführen



Ergebnis speichern



2.3 Grundlegendes Befehlspipelining



Phasen der Befehlsausführung in einer fünfstufigen Pipeline (DLX-Pipeline)

- Befehlsbereitstellungs-Phase (IF-Phase: Instruction Fetch)**
 Der durch den Befehlszähler adressierte Befehl wird aus dem Arbeitsspeicher (bzw. dem Befehlscache) in einen Befehlspuffer geladen. Der Befehlszähler wird weitergeschaltet.
- Dekodier- und Operandenbereitstellungsphase (ID-Phase: Instruction Decode & Register Fetch)**
 Aus dem Operationscode des Maschinenbefehls werden prozessorinterne Steuersignale erzeugt. Die Operanden werden aus (Universal)-Register bereit gestellt (2. Takthälfte).
- Ausführungsphase / Berechnung der effektiven Adresse (EX-Phase: Execution/Effective Address Calculation)**
 Die Operation wird auf den Operanden ausgeführt. Bei Lade- und Speicherbefehlen oder Verzweigungen berechnet die ALU die effektive Adresse.



Phasen der Befehlsausführung in einer fünfstufigen Pipeline (DLX-Pipeline)

Speicherzugriffsphase (MEM-Phase: memory access)

Der Speicherzugriff (bei Lade- und Speicherbefehlen) wird durchgeführt

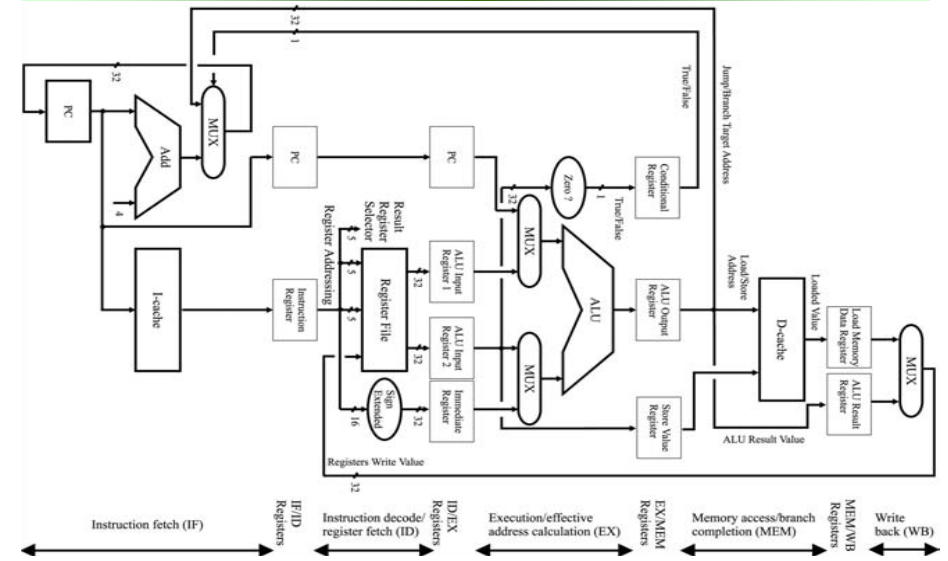
Resultatspeicherphase (WB-Phase: write back):

Das Ergebnis wird in ein (Universal)-Register geschrieben (1. Takthälfte).

Befehle ohne Ergebnis durchlaufen diese Phase passiv.



Pipeline (Übersicht)



Phasen der Befehlsausführung

Befehlsbereitstellungs-Phase (IF-Phase)

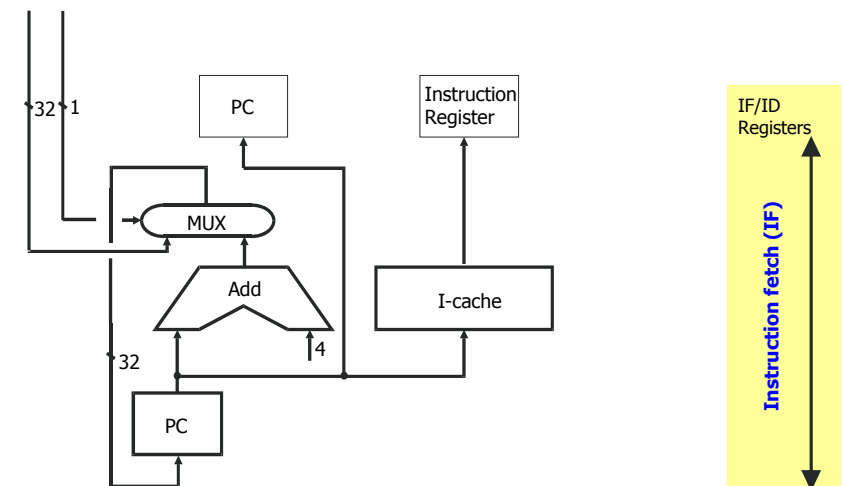
- Der durch den Befehlszähler (PC) adressierte Befehl wird aus dem Arbeitsspeicher bzw. dem Instructioncache (I-Cache) in das Befehlsregister (Instruction Register) geladen.
- Erhöhung des Befehlszählers um 4, um die Befehlsadresse um vier Bytes weiterzuschalten.

Annahme: 32 Bit breites Befehlsformat und Byteadressierung

- Bei vorangegangenen Sprungbefehl kann die Zieladresse aus der MEM-Stufe benutzt werden, um den PC auf die im nächsten Takt zu holende Anweisung zu setzen.



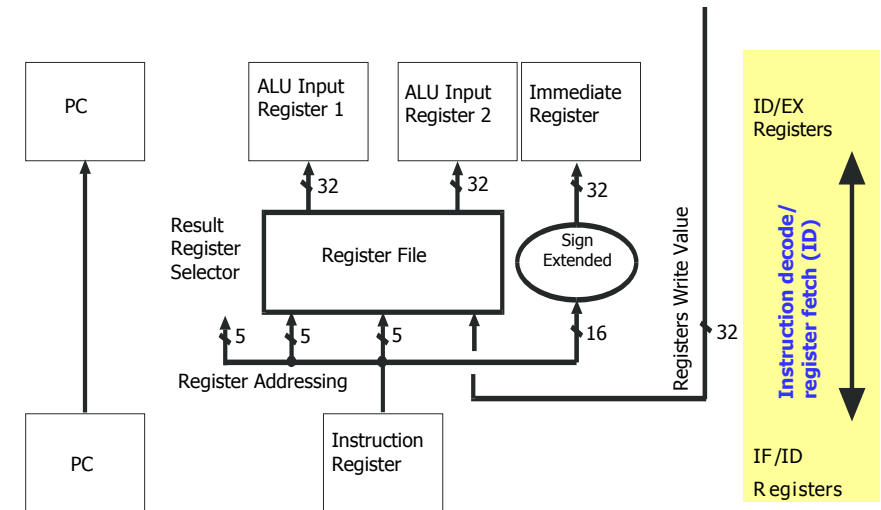
Pipeline (Instruction fetch)



Phasen der Befehlsausführung

- ❑ Dekodier- und Operandenbereitstellungsphase (ID Phase)

- Befehl in der ersten Takthälfte dekodieren
- Abhängig vom Befehlstyp wird eine weitere Aktion ausgeführt:
 - **Register-Register-Befehle** (arithmetisch-logische Befehle):
Laden der Operanden aus den Registern in die ALU-Eingaberegister (ALU Input Register 1 & 2)
 - **Speicherreferenz** (Lade und Speicherbefehle):
Laden eines Teils der Speicheradresse in das erste ALU-Eingaberegister und des vorzeichenenerweiterten Verschiebungswertes in das Verschieberegister (Immediate Register)
 - **Steuerungstransfer** (Sprungbefehle):
Berechnung der Sprungadresse; Auswertung der Sprungbedingung und Speicherung des Ergebnisses (true/false) im Bedingungsregister (Conditional Register)



Phasen der Befehlsausführung

❑ Ausführungsphase (EX Phase)

- Befehl ausführen
- Abhängig vom Befehlstyp wird eine weitere Aktion ausgeführt:
 - **Register-Register-Befehle** (arithmetisch-logische Befehle):
Die ALU führt die Operation auf die Operanden in den Eingaberegistern und legt das Ergebnis im ALU-Ausgaberegister (ALU Output Register)
 - **Speicherreferenz** (Lade und Speicherbefehle):
 - Die ALU berechnet die effektive Adresse aus dem ersten ALU-Eingaberegister und dem Verschiebungsregister und legt das Ergebnis im ALU-Ausgaberegister.
 - Bei einem Speicherbefehl wird der Inhalt des zweiten ALU-Eingaberegister (der zu speichernde Wert) ungeändert in das Speicherwertregister (Load Value Register) verschoben.

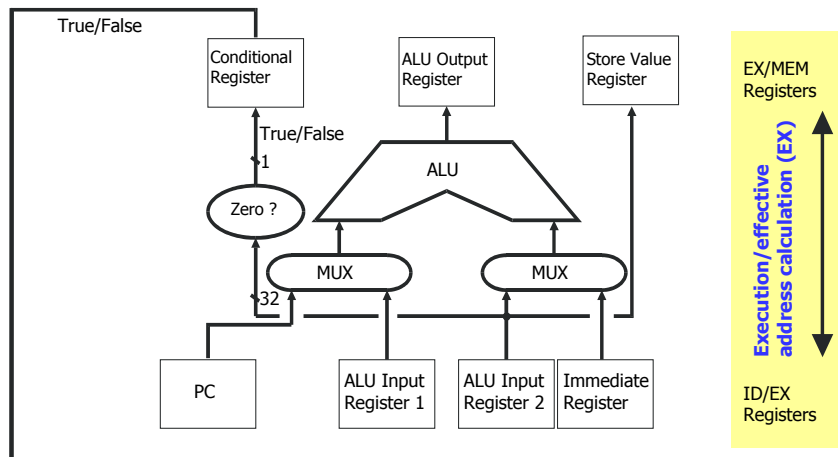
Phasen der Befehlsausführung

❑ Ausführungsphase (EX Phase)

- Befehl ausführen
- Abhängig vom Befehlstyp wird eine weitere Aktion ausgeführt:
 - **Steuerungstransfer** (Sprungbefehle):
 - Die ALU berechnet die Sprungadresse aus dem PC-Register (in der ID/EX-Stufe) und dem Verschiebungsregister und legt die Sprungadresse im ALU-Ausgaberegister ab.
 - Speicherung der Information, ob der Sprung ausgeführt wird oder nicht (true/false) im Bedingungsregister (Conditional Register)

Phasen der Befehlsausführung

❑ **Ausführungsphase / Berechnung der effektiven Adresse (EX-Phase)**



Phasen der Befehlsausführung

- ❑ **Speicherzugriffsphase (MEM-Phase):**

- **Nur für Lade-, Speicher- und bedingte Sprungbefehle**
- Abhängig vom Befehlstyp werden unterschiedlichen Aktionen ausgeführt:

- **Register-Register:**

Das ALU-Ausgaberegister in das ALU-Ergebnisregister übertragen

- **Laden:**

Speicherwort wird mit der von der ALU berechneten effektiven Adresse adressiert, aus dem Daten-Cache gelesen und im Ladewertregister (Load Memory Data Register) geladen

- **Speichern:**

Inhalt des Speicherregisters (Store Value Register) wird in den Daten-Cache geschrieben, wobei der Inhalt des ALU-Ausgaberegister als Adresse benutzt wird.

- **Steuerungstransfer:**

Für genomene Sprünge wird der Inhalt des PC durch den Inhalt des ALU-Ausgaberegister ersetzt, sonst bleibt der Inhalt von PC unverändert.

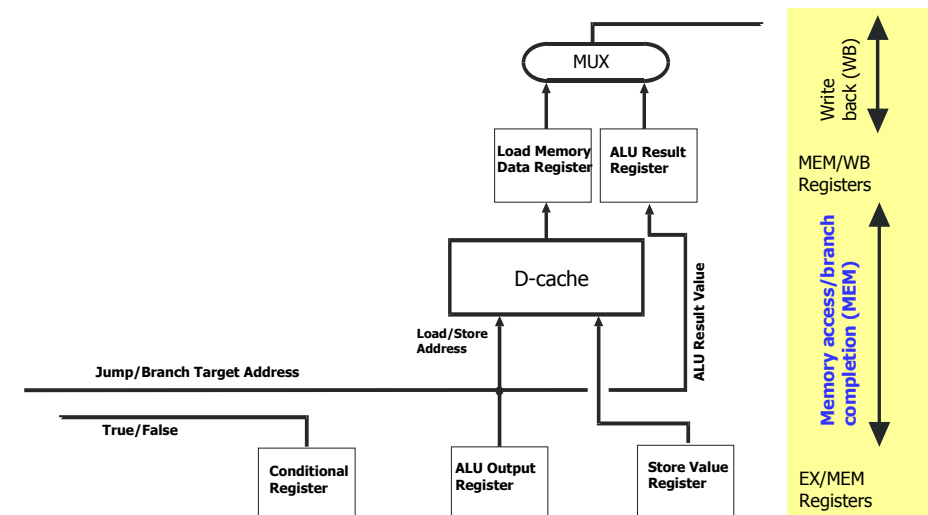
Phasen der Befehlsausführung

❑ Resultatspeicherphase (WB-Phase)

- Während der ersten Takthälfte wird der Inhalt des Ladewertregisters (bei einem Ladebefehl) oder des ALU-Ausgaberegisters (bei allen anderen Befehlen) in ein (Universal)-Register gespeichert
- Das (Universal)-Register, in dem das Resultat geschrieben werden soll wird mit Hilfe von einem Resultatregisterselektor (Result Register Selector) festgelegt.

Bemerkung: Der Resultatregisterselektor wird auch durch die Pipeline weitergegeben (Die Weitergabe im Bild ist nur angedeutet).

Pipeline (MEM&WB)



Diskussion

- Die **Zykluszeit** der Pipeline wird vom **kritischen Pfad** diktiert: der langsamsten Pipeline-Stufe.
- Alle Stufen nutzen verschiedene Ressourcen.
- Im Idealfall wird in jedem Takt ein Befehl in die Pipeline eingefüttert. (CPI=1).

