

CISC = Complex Instruction Set Computer

- mächtige Maschinenbefehle
- umfangreiche Befehle
- viele Befehlsformate / Adressierungsarten

RISC = Reduced Instruction Set Computer

- Befehlssatz besteht aus **wenigen, essentiellen Befehlen** (Anz. ≤ 128) mit einheitlich 32 Bit Befehlslänge
- große Registerzahl, mind. 32 allg. verwendbare Register
- **Register-Register-(Load-/Store-)Architektur**= Lesen und Schreiben nur aus/in Register!
- Beendigung von 1 **Maschinenbefehl pro Prozessortakt**
- 1 Befehlsausführung pro Takt \rightarrow skalarer RISC-Prozessor
- 4 Befehlsausführungen pro Takt \rightarrow 4-fach skalarer RISC-Prozessor

Befehlspipelining = „Fließband-Bearbeitung“

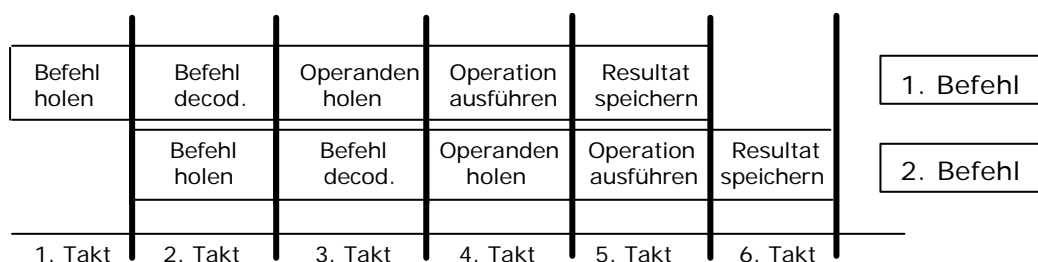
- **Zerlegung** einer Maschinenoperation in mehrere **Teiloperationen**. Diese werden von hintereinandergeschalteten Verarbeitungseinheiten **taktsynchron + taktversetzt** bearbeitet.
- genau **eine Teiloperation pro Verarbeitungseinheit**
- Bearbeitung von **n Befehlen gleichzeitig** (alle sind in unterschiedlichen Phasen)
- pro Proz.Takt wird ein Maschinenbefehl beendet
- Phasen 2 + 3 zusammenfaßbar

Superpipelining

- bei Superpipelining feinere Unterteilung in **mehr als 5 Phasen**
- Befehlsbereitstellungs-Phase und Speicherzugriffs-Phase jeweils unterteilt in:
 - Register-Hol-Phase
 - Tag-Check-Phase (Cache-Tags werden überprüft)
- Addition/Multiplikation auf **Festpunktop operanden** \rightarrow in einem Takt ausführbar
- Addition/Multiplikation auf **Gleitpunktop operanden** \rightarrow drei Takte nötig

Phasen

- **Befehlsbereitstellungs-Phase**
 \rightarrow durch Befehlszähler adress. Befehl aus HS oder CS in Befehlspeicher laden
 \rightarrow Befehlszähler weiterschalten
- **Dekodier-Phase**
 \rightarrow Operationscode des Maschinenbefehls in proz.eigene Steuersignale wandeln
- **Operandenbereitstellungs-Phase**
 \rightarrow Operanden aus Register holen
- **Ausführungs-Phase**
 \rightarrow ALU-Operation auf Operanden ausführen
- **Resultatspeicher-Phase**
 \rightarrow Ergebnis in Register speichern



Struktur skalarer μP

-

Struktur superskalärer μP

- ~10 unabhängig + parallel arbeitende Ausführungseinheiten(AE)
- 1 superskalare Befehlszuordnungseinheit kann pro Takt den AE 2 Befehle zuordnen
- Verzweigungseinheit zur statischen Sprungvorhersage; wird von Sprung-Ziel-Cache unterstützt (Speicherung von Sprungbefehlen und deren Zielbefehlen)
- Datenspeicher (8Kb)
- CodeCache-Speicher (8Kb)
- zwei Registersätze:
 - 32 allg. Register (32 Bit breit)
 - 32 Gleitpunkt-Register (80 Bit breit)

superskalare Befehlspipeline

- ist reine **Implementierungstechnik**, unabhängig von Architektur
- Voraussetzung: mehrere **parallel arbeitende AE** vorhanden
- superskalare Prozessoren können **mehrere Befehle pro Takt** gleichzeitig den AE zuordnen und
- **dynamische** Zuordnung von Befehlen durch Befehlszuordnungseinheit (per HW, nicht durch Compiler)
- Befehle werden **in Programmreihenfolge** zugeordnet
- Befehlausführung durch Verzögerungen nicht in Progr.reihenfolge : durch **Reservation Stations** (= spez. Befehlspuffer) verhinderbar
- Befehlsbeendigung nicht in Progr.reihenfolge → Behebung durch Rückordnungspuffer (FIFO - Puffer)

Vergleich Superskalar-/Superpipelining-Technik

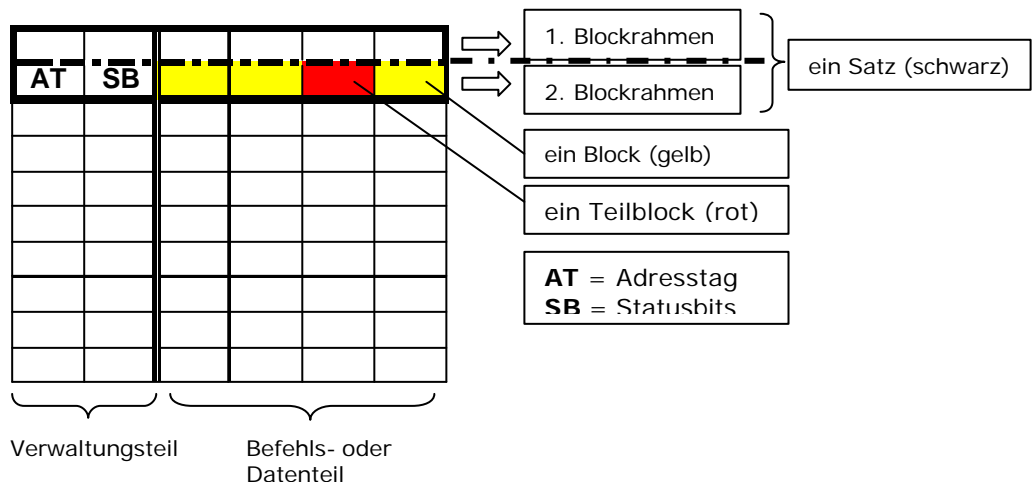
- Superskalar-Technik betont **räumliche Parallelität** (mehrere Op. parallel auf unterschiedlichen AE)
→ **mehr Transistoren** wichtig
- Superpipelining-Technik betont **zeitliche Parallelität** (mehrere Op. auf einer AE überlappend)
→ schnellere Taktrate in den Pipelinestufen wichtig → **schnellere Transistoren** wichtig
- Anzahl der Transistoren auf Chip wächst schneller als deren Geschwindigkeit
- Kombinationen

Cache-Speicher (CS)

- CS = kleiner, schneller Pufferspeicher
- beinhaltet i.d.R. die **Daten(-Kopien)** derjenigen Teile des HS, auf die der Prozessor wahrscheinlich als nächstes zugreifen wird
- erwünscht: CS-Zugriff fast so schnell wie Registerzugriff
 - CS als **Primary Level Cache** direkt auf dem Prozessorchip
 - CS als **Secondary Level Cache** in Form von SRAM realisiert
- **Nutzen:** Verringerung der mittleren Zugriffszeit auf HS
- CS-**Verwaltung** in HW realisiert
- Cache-**Controller** kopiert Daten in Cache
- sogenannte **Cache-Treffer** ↔ **Cache-Fehlzugriffe**

CS-Organisation

- CS besteht aus mehreren **Sets** (alle Sets == Menge d. Blockrahmen im CS)
- ein Set besteht aus mehreren **Blockrahmen** → Anzahl der Blockrahmen in einem Satz ist die Satz-Assoziativität
- in einem Blockrahmen steht ein **Block** [Block = die Datenportion, die in einen Blockrahmen paßt; ist gleichzeitig die Datenportion, die zw. CS und HS übertragen wird]
- Ein Block besteht aus mehreren **Teilblöcken**
- **Blocklänge** = Anzahl d. Speicherplätze
- **Gesamtzahl der Blockrahmen = Anzahl der Sätze x Satz-Assoziativität** , $C = S \times N$
- Verwaltungsteil besteht aus:
 - **Adressbit** (enthält entweder physik. HS -Adresse oder logische Adresse)
 - **Statusbits** (Datenkopie gültig/ungültig)
- CS heißt:
 - vollasoziativ wenn $S=1, N=C$ (CS besteht aus nur 1 Satz)
 - direkt-abgebildet wenn $N=1, S=C$ (jeder Satz enthält nur 1 Blockrahmen)
 - n-fach satzassoziativ wenn $S=C/N$ (in allen anderen Fällen)



1. Rechenbeispiel

- gegeben
 - CS-Größe von 2 KB
 - Block-Größe von 16 B

- 4-fach Satzassoziativität
- CPU versucht 1026-stes Doppelwort (1 Wort = 1 Bit , 1 DW = 2 Bit) zu adressieren
- gesucht
 1. Anzahl der Blöcke
 2. Anzahl der Sätze
 3. Bestimmung von Satznummer und Adresstag
- Umschreibung ::

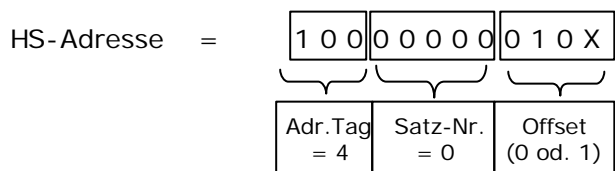
| | | | | |
|--------|--------|---|----------|------|
| CS: | 2KB | = | 2^{11} | Byte |
| Block: | 16Byte | = | 2^4 | Byte |
| S.a.: | 4 | = | 2^2 | |

zu 1) CS-Größe = Anz. d. Blöcke * Block-Größe
 2^{11} Byte = Anz. d. Blöcke * 2^4 Byte
 Anz. d. Blöcke = $2^{11-4} = 2^7$

zu 2) Satzass. * Anz. d. Sätze = Anz. d. Blöcke
 2^2 * Anz. d. Sätze = 2^7
 Anz. d. Sätze = 2^5

zu 3) 1026 Bytes = 1024 + 2 Bytes = $2^{10} + 2^1$
 1026-stes DW = (1024 + 2 Bytes)*2 = $(2^{10} + 2^1) * 2$
 = $2^{11} + 2^2$

1024 Byte = 1 0 0 0 0 0 0 0 0 0 0 0
 + 2 Byte = 0 0 0 0 0 0 0 0 0 0 1 0



2. Rechenbeispiel

- gegeben
 - CS-Größe von 16 KB
 - Block-Größe von 8 B
 - 4-fach Satzassoziativität
 - CPU versucht 717-stes Doppelwort (1 Wort = 1 Bit , 1 DW = 2 Bit) zu adressieren
- gesucht
 1. Anzahl der Blöcke
 2. Anzahl der Sätze
 3. Bestimmung von Satznummer und Adresstag
- Umschreibung ::

| | | | | | | |
|--------|--------|---|----------------|------|----------|------|
| CS: | 16 KB | = | $2^4 + 2^{10}$ | = | 2^{14} | Byte |
| Block: | 8 Byte | = | 2^3 | Byte | | |
| S.a.: | 4 | = | 2^2 | | | |

zu 1) CS-Größe = Anz. d. Blöcke * Block-Größe
 2^{14} = Anz. d. Blöcke * 2^3
 Anz. d. Blöcke = $2^{14}/2^3 = 2^{14-3} = 2^{11}$

zu 2) Satzass. * Anz. d. Sätze = Anz. d. Blöcke
 2^2 * Anz. d. Sätze = 2^{11}
 Anz. d. Sätze = $2^{11}/2^2 = 2^{11-2} = 2^9$