

## Microarchitecture

- **CPU** (Abb. siehe Folie 2.3ff)
  - **Steuerwerk (CU)**
    - Befehlsregister (enthält aktuellen Befehl), Dekoder (mikroprogrammiertes Schaltwerk), Taktgenerator (externer Quarz)
    - **Synchrones dynamisches Schaltwerk**, Speicherung in Kondensatoren anstatt Flipflops (→ **Mindesttaktfrequenz erforderlich**, sonst gehen Kondensatorinhalte durch Leckströme vor dem nächsten Taktzyklus verloren)
    - **CISC: Mikroprogrammiersteuerwerk** (Mikroprogramm = Folge von Mikrobefehlen), nicht änderbar durch Benutzer, mehrere Befehlsregister für unterschiedlich lange Befehlsformate und Opcode-Prefetching, **RISC: festverdrahtetes Schaltwerk**
    - Statusregister/Steuerregister (Interrupt enable Bit, User/System Bit, Trace Bit, Decimal Bit)
    - Interpretation Maschinenbefehle unter Berücksichtigung der Statusinformation in Steuerbefehle
  - **Rechenwerk (Operationswerk)**
    - ALU als Schaltnetz mit Steuereingängen und Statusregister (Carry, Overflow, Zero, Sign, ...)
    - Operationen: Arithmetisch, Logisch, Schieben, Rotation, Transport
    - Akkumulatoren aus Akkumulator-Register und Hilfsregister, heute Akkumulator im Registersatz
    - Statusregister (CCR) und Steuerregister zusammengefasst als **Prozessorstatuswort (PSW)**
  - **Registersatz**
    - Erweiterung des Operationswerks, schneller Zwischenspeicher (kleine Zugriffszeit)
    - General Purpose Register (Daten- und Adressregister) und Special Purpose Register (PC, PSW, SP)
    - Modifikation von Indexregister (Post-Inkrement, Pre-Dekrement, Auto-Inkrement, Auto-Dekrement)
  - **Adresswerk (AU)**
    - Adressenberechnung inkl. MMU
  - **Systembus-Schnittstelle (BIU)**
    - Zwischenspeicher für PC, AP, DBP, I/O-Treiber (Tristate-Treiber)
  - **Internes Bussystem**
    - höhere Parallelität und Geschwindigkeit durch Aufspalten des internen Datenbus in Teilbusse (Busse erfordern erhebliche Chipfläche)
    - **Prefetch Bus** (zweiter getrennter Datenbus zum Adress-/ Steuerwerk)
    - **Ergebnis Bus** (zusätzlicher Bus, der Ergebnisse des Rechenwerks in den Registersatz transportiert)
    - **Zweiter Operandenbus** (beide Operanden parallel an die ALU)
- **Leistungssteigerung in Rechnersystemen**
  - **Technologische Maßnahmen** (schnellere Technologien, Redesign nötig)
  - **Strukturelle Maßnahmen** (Transistorzahl erhöhen, Parallelarbeit)
  - Klassifizierung von Rechnerstrukturen nach Flynn (*Folie 2.103ff*)
    - **SISD** (ein Datenstrom entsprechend einer seriellen Befehlsfolge verarbeitet (von-Neumann-Rechner))
    - **SIMD** (alle Prozessoren gleichzeitig dieselben Befehle auf versch. Daten (Array-Prozessoren))
    - **MIMD** (alle Prozessoren gleichzeitig versch. Befehle auf versch. Daten (Multiprozessor PC))
    - **MISD** (ein Datenstrom bearbeitet, Ausführungseinheiten übernehmen Teiloperationen (Pipeline-Verarbeitung), dadurch Parallelität auf Befehlsebene (Intel 80286))
  - **Pipelining** (Zerlegung einer Operationen in mehrere Phasen, hintereinander taktsynchron bearbeitet)
  - im Idealfall pro Taktzyklus ein Befehl, jedoch **Pipeline-Hemmnisse** möglich durch **Daten- und Kontrollabhängigkeiten** (→ Forwarding, Reordering, Branch Prediction)

## Instruction Set Architecture

- **ISA als Schnittstelle zwischen Hardware und Software**
- **CISC (Gründe und Prinzipien)**
  - Geschwindigkeitsunterschiede zwischen CPU und Hauptspeicher → komplexe Befehle → kompakter Code
  - Mikroprogrammierung, Unterstützung höherer Programmiersprachen (Direkte Abbildung)
  - Abwärtskompatibilität, Marktstrategie (Lock-In, Wechselkosten, Umfang des Befehlssatz oft als Werbegrund)
  - Unterstützung von Compiler durch entspr. Befehle, Unterstützung spezieller Einsatzgebiete (MMX)
  - **Entwicklung von Hardware, Programmiersprachen und Einsatzgebieten begünstigt „komplizierte“ Befehle**
- **CISC (Kritik und Grenzen)**
  - Schnellere Hauptspeicher und Cache beschleunigen Programmausführung, Ausnutzung nur eines kleinen Teils des Befehlssatzes (~ 20%)
  - verlängerte Entwurfszeit, sehr komplexe Steuerwerke, umfangreiche Mikroprogramme, Fehlerhäufigkeit auf Mikroprogrammebene, schwieriger Compilerbau
- **RISC (Gründe und Prinzipien)**
  - vielbenutzte Befehle so schnell wie möglich ((super-)skalar, Befehls-Pipeline), größte Teil der Arbeit durch optimierte Compiler zur Übersetzungszeit
  - Operanden nach Möglichkeit in großen Registersätzen (schneller Zugriff/Verarbeitung), einheitliche Befehlsformate (schnelle Dekodierung), nur Load/Store-Architektur (weniger Adressierung, schnelle Ausführung), Pipelining so gut es geht
  - Coprozessor für komplexe Befehle
  - Implementierung auf einem Chip, kurze Entwurfszeiten, hohe Taktraten, Pipelining, eingesparte Chipfläche

- meist schneller als CISC durch **vollständige Parallelarbeit aller Komponenten** (extreme Pipelining-Verarbeitung), mit großer Wahrscheinlichkeit **in jedem Taktzyklus ein Befehl** beendet
- **skalare RISC-Prozessoren** (durchschnittlich eine Befehlsausführung pro Takt)
- **superskalare RISC-Prozessoren** (durchschnittlich vier oder mehr Befehle pro Takt)
- **Havard-Architektur** (getrennter Programm- und Datenspeicher → zwei Adress- und Datenbusse → paralleles Holen von Operanden und Instruktionen)
- **Adressierungsarten**
  - Adressberechnung dynamisch in Effektive Logische Adresse, durch MMU in physikalische Adresse
  - **Register-Adressierung** (Operand steht bereits im Register)
    - **implizit** (EA codiert im OpCode enthalten, z.B. MOV EAX, EBX, speziell Flag-Adressierung, z.B. STC)
    - **explizit** (EA im Operandenfeld des Befehls angegeben, z.B. DEC EAX)
  - **Speicher-Adressierung** (eine Adressberechnung zur Ermittlung der EA)
    - **Unmittelbar** (Befehl enthält Operanden selbst, z.B. MOV EAX, 14)
    - **Direkt-absolut** (Befehl enthält im Speicherwort EA, keine weiteren Manipulation, z.B. JMP \$07FE, Spezialfall: Zero-Page und Seiten-Register)
    - **Register-indirekt** (EA im Registerfeld des OpCodes angegebenen Adressregisters (Zeigeradressierung), z.B. MOV [EAX], 14 (postincrement oder predecrement möglich, Offset und Basisindex möglich))
    - **Speicher-indirekt** (EA Speicherinhalt des Operanden als Speicheradresse, z.B. MOV EAX, [07FE])
- **Befehlsarten**
  - Transferbefehle (LEA, MOV, PUSH, POP)
  - Ein-/Ausgabebefehle
  - Arithmetische Befehle (ADD, SUB, MUL, DIV, CMP, INC, DEC)
  - Logische Befehle (AND, OR, NOT)
  - Flag- und Bit-Manipulationsbefehle (STC, CLC, BT)
  - Schiebe- und Rotationsbefehle (SHL, SHR, SAL, SAR)
  - Befehle zur Programm-Steuerung (Spring- und Verzweigungsbefehle: JMP, Jxx, Unterprogramm- und Rück-Sprünge, Software-Interrupts: CALL, RET, SWI, RTI)
  - Systembefehle (NOP, HALT, RESET)
  - Zeichenketten- und Blockbefehle
  - Zusammengesetzte Befehle (LOOP, BTC)
- **Nicht-lineare Programmausführung**
  - bedingt durch Sprünge, Verzweigungen, Prozeduraufrufe, Sub-Routinen (*siehe Folie 3.89*), Multithreading, Co-Routinen (*siehe Folie 3.90*), HW-Interrupts, SW-Interrupts, Traps, usw.
- **Interruptbehandlungen**
  - Exceptions: Fehler im Betriebssystem bei Ausführung des Anwenderprogramms oder Fehler der Hardware, oder Wunsch externer Systemkomponenten, die Aufmerksamkeit des Prozessors zu erhalten
  - Auswahl und Aktivierung der ISR durch Steuerwerk (Interrupt System), Ähnlichkeit mit Unterprogrammaufruf
    - CALL bei Unterprogramm, INT bei Interrupt
    - RET bei Unterprogramm, RETI bei Interrupt
    - Einsprungadresse bei Unterprogramm direkt im Programm, bei Interrupt über Interrupt-Vektor-Tabelle
    - Unterprogramm sichert meist nur PC auf Stack, Interrupt sichert meist PSW
    - Unterprogramm immer ausgeführt, Interrupts nur wenn Interrupt Enable Bit im PSW gesetzt
  - **Prozessorexterne Ursachen** (asynchrone Ereignisse), z.B. RESET, HALT, ERROR, Interrupt
  - **Prozessorinterne Ursachen** (synchrone Ereignisse), z.B. Software Interrupts, Traps
  - Behandlung mehrerer Interrupt-Quellen
    - ICR fragt zyklisch nach Interrupt-Flag, falls gesetzt → Abfrage abbrechen, ISR starten, weitere Interrupt-Anforderungen möglich während der Abarbeitung
    - **Polling (1. Variante)**: Zyklische Abfrage während ISR fortgesetzt („faire“ Prozessor-Zuteilung)
    - **Polling (2. Variante)**: Zyklische Abfrage beginnt immer mit erster Komponente (Prioritäten zugeordnet)
    - Nachteile: Priorisierung und Identifizierung durch zyklische Abfrage (Software) sehr zeitaufwendig
    - **Daisy-Chain-Verfahren**: Dezentrale Priorisierung mittels Priorisierungsschaltung (Prioritätskette)

## Operating System Architecture

- Betriebssystem umfasst Programme, die die Grundlage der möglichen Betriebsarten bilden und insbesondere die Abwicklung von Programmen steuern und überwachen
- Betriebssystem soll **semantische Lücke** zwischen Anwendung (möglichst hohe Operationen) und Hardware (nur elementare Operationen) zu **verkleinern**
- **Ressourcenverwaltung** (Betriebsmittelverwaltung, resource management)
  - einfache, effektive und konfliktfreie Nutzung der Betriebsmittel durch die verschiedenen Prozesse
  - Dateien (Sequenz von Bits), Datenstrukturen (File Tables, Ordner)
  - Neue Instruktionen für I/O in ISA (create, unlink, open, close, CreateFile, CloseHandle...)
- **Prozessverwaltung** (Auftragesverwaltung, task management)
  - Verwaltung der Prozesszustände (aktiv, blockiert, bereit)
  - möglichst effiziente und scheinbar parallele Bearbeitung mehrerer Aufgaben
  - Prozess Koordination (z.B. semaphores) und Kommunikation (IPC)
- **Speicherverwaltung** (memory management)

- verfügbaren Arbeitsspeicher virtuell vergrößern (benötigten Teile im Arbeitsspeicher, Rest im Hintergrundspeicher bei Bedarf geladen (**swapping, paging**))
- Verwaltung des virtuellen Speichers durch MMU (**Umsetzung virtueller (logischer) Adressen in physikalische Adressen**)
- **Seitenwechself Verfahren**
  - logische und physikalischer Adressraum in „Segmente fester Länge“ (pages)
  - **Vorteile:** durch kleine Seiten wirklich benötigter Programmteil eingelagert, geringerer Verwaltungsaufwand als Segmentierung
  - **Nachteile:** häufiger Datentransfer
- **Segmentierungsverfahren**
  - Segmente verschiedener Länge mit logisch zusammenhängenden Informationen
  - **Vorteile:** spiegelt logische Programmstruktur wieder, relativ seltener Datentransfer, segmentspezifische Schutzmaßnahmen möglich
  - **Nachteile:** wenn Datentransfer, dann jedoch umfangreich, besteht Programm nur aus Code- oder Daten-Segment, so muss es vollständig eingelagert werden
- **Einlagerung**
  - **Demand Paging** (Daten eingelagert, sobald auf sie zugegriffen wird, **Segment- oder Seitenfehler** (segment fault, page fault) bei Zugriff auf nicht vorhandenes Segment oder Seite)
  - Erkennung Segmentfehler durch Bit im Segment-Deskriptor, Erkennung Seitenfehler durch Seitentabelle oder spezielles Kennungsbit im Seitentabellen-Verzeichnis (Übersetzungstabellen)
- **Zuweisung**
  - bei Segmentierung: first-fit (erste passende Lücke), best-fit (kleinste passende Lücke) oder worst-fit (größte passende Lücke) → **externe Fragmentierung**
  - bei Seitenwechsel: keine externe Fragmentierung, aber durch einheitliche Seitengröße hohe Wahrscheinlichkeit für ungenutzten Leerraum auf der letzten Seite → **interne Fragmentierung**
- **Ersetzung**
  - bevorzugt unveränderte Seiten ersetzen, weil kein Rückschreiben nötig
  - **FIFO (first-in-first-out):** geeignet für lineare Ausführungen
  - **LIFO (last-in-first-out):** geeignet für einmaligen Aufruf innerhalb einer Schleife
  - **LRU (least recently used):** hier Zugriffszeit durch Stack ermittelbar
  - **LFU (least frequently used):** Zähler benötigt, Nachteil, wenn Aufruf anfangs sehr häufig benutzt und später gar nicht mehr
  - **LRD (least reference density):** Gesamtzahl der zugriffe im Einlagerungszeitraum/(Gesamtzahl aller Referenzen – Gesamtzahl der Referenzen zum Zeitpunkt der Einlagerung)
- **Lokalitätseigenschaften**
  - Programme greifen in einem kleinen Zeitintervall auf einen relativ kleinen Teil des Adressraumes zu
  - **Zeitliche Lokalität** (baldige Wieder-Referenzierung nach Referenzierung, z.B. bei for-Schleifen)
  - **Örtliche Lokalität** (baldige Wieder-Referenzierung mit benachbarten Adressen, z.B. bei Arrays)
- **Cache-Einbindung**
  - **Virtueller Cache** (zwischen CPU und MMU, Speichern von virtuellen Adressen)
  - Vorteil: bei Treffern wird die MMU nicht benötigt
  - **Physikalischer Cache** (zwischen MMU und Speicher, Speicher von physikalischen Adressen)
  - Vorteil: physikalische Adresse kleiner als virtuelle (weniger Bits zu speichern), falls MMU auf Prozessorchip nur physikalischer Cache außerhalb erweiterbar
- **Schutzmechanismen**
  - Trennung der Systemsoftware von den Anwendungsprozessen
  - Trennung der Anwendungsprozessen voneinander (Schutzebenen und Zugriffsrechte)
- **Kernelarten**
  - **Monolithischer Kernel** (fast alle Funktionen direkt im Kern eingebaut, z.B. UNIX)
  - **Mikrokern** (Grundfunktionen im Kern, Gerätetreiber auf Nutzerebene, z.B. Windows NT)

### Assembly Language Level

- **Assembler**
  - Umwandlung der symbolische Repräsentation der Maschinensprache (Assemblersprache) in Numerische Repräsentation der Maschinensprache (ISA-Instruktionen)
- **Compiler**
  - Umwandlung einer Hochsprache (Java, C, C++,...) in Assemblersprache oder auch mit built-in-assembler direkt in ISA-Instruktionen
- **Assemblersprache**
  - 1:1 Darstellung der Befehle in ISA-Instruktionen mit zusätzlichen symbolischen Namen, Adressen, Labels, usw.
- Assembler geeignet für vollständigen Zugriff auf Hardwarefeatures (Register, Flags, usw.) und Performancegewinn durch optimierten Code für spezielle Anwendungen, aber schwer zu programmieren
- **Assemblierungsprozess**
  - Schritt-für-Schritt funktioniert nicht (Symbole genutzt vor Definition)
  - **Two-Pass Translator oder Single-Pass plus Conversion into intermediate format** (1. Schritt: Syntax überprüfen, Symboltabelle, Operationstabelle, usw. erstellen, Instruktionslängen überprüfen, 2. Schritt: Object code generieren und Informationen für Linker generieren → Linking and Relocation)
- **DLL**

- Sammlung von Programmfunktionalitäten (Hilfsmodule) für zusammengehörende Aufgaben
- reduzieren Speicherverbrauch, da bei Bedarf nur **einmal** in Arbeitsspeicher geladen werden müssen
- viele Prozesse „teilen“ sich den selben Code / Instruktionen → Vorteil für Multitasking-Systeme

### Computer Systems Organization

- **Architektur klassischer PC-Systeme** (Abb. siehe Folie 6.6)
- **Chipsatz**
  - Bindeglied zwischen einzelnen Komponenten, Leistungsunterschiede bis zu 10%
  - legt fest, welche Komponenten verwendet werden können (Systembus, Speichertyp, Schnittstellen, Prozessortyp)
  - Northbridge (synchronisiert Datentransfer und Datensteuerung zwischen CPU, Arbeitsspeicher, Cache und AGP)
  - Southbridge (Datentransfer und Datensteuerung zwischen peripheren Geräten (PCI-Bus, ISA-Bus, ATA...) und weiteren Schnittstellen)
- **Gehäuse und Anschlüsse**
  - Gehäuse zur mechanischen Stabilität, Herausführung der Anschlüsse, Ableitung der Wärme (meist Plastik oder Keramik)
  - **DIP**
    - zwei parallele Reihen Anschlussstifte (Pins) von 8 bis 64, Chip in der Mitte des Gehäuses
  - **Quad-Pack-Gehäuse**
    - einreihige Anschlüsse an jeder Gehäuseseite
    - **LCC** (einfache Kontakte an Gehäuseseite → spezielle Sockel mit Federkontakten zur Montage)
    - **Pins** (konventionelle Pins an Gehäuseseite)
    - **SMD** (Lötflächen zur direkten Oberflächenmontage)
  - **PGA/BGA**
    - Matrixförmige Pin-Anordnung in Form eines Nagelbretts bis ca. 190 Pins, BGA mit bis zu 625 Pins
- **Mikroprozessor-Anschlüsse**
  - **Stromversorgung**
  - **Systemtakt**
  - **Adressbus** (reine Ausgangssignale, Quelle und Ziel eines Datentransports, z.B. Speicher, Peripherie, usw.)
  - **Datenbus** (Daten- und Befehls-transport, bidirektional)
  - **Steuerbus** (Steuern von Systemkomponenten und Melden von Systemzuständen, unidirektional)
    - Daten-/Adressbus-Steuersignale, System-Steuersignale, Prozessor-Steuersignale, Unterbrechungssignale
    - Signale nach internationalem **TTL-Pegel**
- **Busse**
  - **Master** (aktiver Knoten, der auf den Bus zugreifen kann), **Slave** (passiver Knoten)
  - **Bus-Arbitr** regelt Hierarchie der Zugriffsberechtigungen und gewährt Zugriff auf Systembus
    - **Zentrales Verfahren** (unabhängige Anforderung, Zugriffsforderung von zentralen Prioritäts-Decoder)
    - **Dezentrales Verfahren (Daisy-Chain)**
  - **Synchrone Busse**
    - alle Vorgänge (Lesen/Schreiben) laufen synchron zum Takt nach einem starren Muster
    - synchroner Bus mit einer festen Nummer an Lese-/Schreibzyklen
  - **Semi-Synchrone Busse (Intel, Motorola 68040)**
    - moderne Prozessoren haben höhere Taktfrequenz → Schreibe-/Lesezyklus braucht mehrere Taktzyklen
    - Steuereingang (READY) zur Anforderung von Wartezyklen (wait states)
    - Bus immer noch synchron (streng am Takt orientiert), Dauer eines Buszyklus jedoch als Vielfaches von Taktzyklen variierbar
    - höherer Steueraufwand als synchroner Systembus, Zeitverhalten auf versch. schnelle Bausteine anpassbar, bei ausreichend schnellen Bausteinen kann READY fest auf 0 gelegt werden
  - **Asynchrone Busse (Motorola 68000 – 68030)**
    - Zeitliche Abläufe durch **Handshake-Signale** (AS, DTACK) gesteuert
    - Anschluss von Komponenten mit fast beliebiger Laufzeit möglich
    - Systemtakt spielt keine Rolle mehr für Synchronisation der Übertragung
  - **Multiplexbusse (Zeitmultiplexverfahren TDM)**
    - benötigt, falls bestimmte Gruppen von Signalen zeitlich hintereinander über dieselbe Busleitung geschickt werden müssen, dadurch Einsparung von Busleitungen (z.B. PCI-Bus als gemeinsamer Daten- und Adressbus)
    - Adresse muss während der gesamten Zugriffszeit vorliegen → Speichern in Flipflops (Latches)
    - Blocktransfer von Daten möglich → konstante Datenübertragungsrate
  - **PCI** (siehe Folie 6.63)
  - **USB**
    - Peripherieverbindung für niedrige Datenraten, Plug-and-Play, Isochroner Datentransfer
    - Baumstruktur über Hubs, serielle Übertragung mittels timeframes
- **Arbeitsspeicher**
  - Gedächtnis (memory) des Computers, Programme und Daten gespeichert, die „**jeder Zeit sofort**“ zur Verfügung stehen müssen
  - permanente Ablage (Festwertspeicher ROM, nicht flüchtig, z.B. Kernel, Systemtabellen)

- vorübergehende Ablage (Schreib/Lesespeicher RAM, flüchtig, z.B. Anwenderprogramme)
- **Speicherelement** (1 Bit Speicher)
- **Speicherzelle (-platz, -stelle)** (feste Anzahl von Speicherelementen, durch eine Adresse ausgewählt, z.B. 8, 16, 32 Bit)
- **Speicherwort** (maximale Anzahl von Speicherelementen, die in einem Buszyklus zwischen CPU und Speicher übertragen werden können → Speicherwortbreite = Datenbusbreite)
- **wahlfreier Zugriff** (Speicherzelle kann direkt angesprochen werden, Selektion über Adressdekoder)
- **Organisation** (n x m Bit-Speicher, n = Speicherzellen, m = Speicherelemente/Zelle)
- **Kapazität** (Informationsmenge in Bit, die im Speicher untergebracht werden kann = n \* m)
- **Arbeitsgeschwindigkeit**
  - **Zugriffszeit (access time)** (maximale Zeitdauer vom Anlegen einer Adresse an den Speicher bis Ausgabe der gewünschten Daten)
  - **Zykluszeit (cycle time)** (minimale Zeitdauer zwischen zwei hintereinander folgenden Aufschaltungen von Adressen an den Speicher)
  - Zykluszeit meist erheblich länger als Zugriffszeit (bis zu 80%), Idealfall: Zykluszeit = Zugriffszeit
  - Speicher muss sich „erholen“, Speicherinformation durch Auslesen zerstört und muss wieder eingeschrieben werden (refresh)
- **Klassifizierung von Halbleiterspeicher** (siehe Folie 6.79)
- **DRAM**
  - behauptet sich als Arbeitsspeicher Nr. 1, größte Integrationsdichte aller Halbleiterspeicher, bitweise organisiert, getrennter Datenein- und -Ausgang
  - Speicheradressen gemultiplext, Auswahl der Spalten- bzw. Zeilenadressen über RAS und CAS
  - Seitenzugriff beschleunigt Zugriff (Zeilenadresse einmal angelegt und gespeichert, Spaltenadressen in schneller Folge angelegt (page mode) → FPM-DRAM (fast page mode))
- **EDO-DRAM**
  - FPM-DRAM um ein Latch-Speicher erweitert, CPU kann Daten auslesen während Speichersteuerung eine neue Speicheradresse an das DRAM übergibt → Art „Pipelining“ → höherer Datendurchsatz
  - genauso wie FPM-DRAM asynchron mit Handshaking-Verfahren
- **SDRAM**
  - es können gleichzeitig zwei Speicherseiten geöffnet werden
  - Verzicht auf Flankensteuerung, Steuerbefehle als 4 Bit Muster in 4-stufiger Pipeline
  - synchron zum Systemtakt
- **DDR-SDRAM**
  - entsprechen SDRAM, Speicherzellen zweimal pro Takt ausgelesen
  - Nutzung beider Taktflanken → doppelter Datendurchsatz
- **Speichermodule-Typen**
  - **Single Inline Pin Package (SIPP)**: 30 pin-förmige Anschlüsse, Datenbreite von 8 Bit
  - **Single Inline Memory Module (SIMM)**: PS/2-Module, 30- und 72-polige Ausführung, Datenbreite von 8 und 32 Bit
  - **Dual Inline Memory Module (DIMM)**: 168-polig, Datenbreite von 64 Bit
  - **Rambus Inline Memory Module (RIMM)**: 400-800 Mhz möglich, Kühlung durch Metallabdeckung
- **Organisation des Arbeitsspeichers** (siehe Folie 6.108 und 6.110)
  - lineare Liste von Speicherworten, aufgebaut aus Speicherbausteinen
  - Zugriffszeit abhängig von der Art der verwendeten Bausteine, Datenbreite entspricht Datenbusbreite
  - **Big Endian** (höchstwertigstes Byte an der niedrigsten Speicheradresse, SPARC, IBM)
  - **Little Endian** (niederwertigstes Byte an der niedrigsten Speicheradresse, Intel)
  - Speicherkapazität auch bei breiteren Organisationsformen (> 8 Bit) in Bytes angegeben
- **Speicherbelegungsplan (Memory Map)**
  - gibt an, welche Speicherbausteine für welche Bereiche des Arbeitsspeichers verwendet werden (z.B. Trennung von ROM, IO, SRAM und DRAM)
  - Speicherbreite variiert über Speicherbereich
- **Modularer Speicheraufbau**
  - Speicher auf Steckkarten verteilt, über Grundplatine mit Systembus verbunden → Erweiterbarkeit → anstelle zentraler Adressauswahllogik nun dezentrale Adressauswahllogik erforderlich
  - Unterteilung der Bus einer Speicheradresse: Karte – Modul – Baustein – Speicherzelle
- **Speicherhierarchie** (siehe Folie 6.125)
  - Ein technologisch einheitlicher Speicher mit **kurzer Zugriffszeit und großer Kapazität** ist aus **Kostengründen** i. A. nicht realisierbar
  - Lösung: **Schichtenweise Anordnung verschiedener Speicher und Verschiebung der Information zwischen den Schichten (Speicherhierarchie)**
    - **Cache-Speicher** (kurze Zugriffszeiten → Beschleunigung des Prozessorzugriffs)
    - **Virtueller Speicher** (Vergrößerung des tatsächlich vorhandenen Hauptspeichers)
  - Wirkung: wie ein großer und schneller Speicher, wenn
    - Lokalitätsverhalten der Programmverarbeitung
    - Umlagerung der Information rechtzeitig (Umlagerungsplan)
    - Inhomogenität des Speichersystems nicht sichtbar (Virtueller Speicher)
  - Leistungsfähigkeit der Hierarchie bestimmt durch Eigenschaften der Speichertechnologie (Zugriffsart, Zugriffszeit), Adressierung der Speicherplätze und Organisation des Betriebs
- **Cache-Speicher**

- schneller kleiner Pufferspeicher, der von größeren langsameren Speicher geschaltet wird um dessen Zugriffszeit zu vermindern
- Buszykluszeit moderner Prozessoren kürzer als Zykluszeit des DRAM → erzwingt Wartezyklen
- SRAM als Alternative klein, teuer, höhere Verlustleistung → nur relativ kleine Speicher möglich
- zwischen Prozessor und DRAM legt man kleinen schnellen SRAM (Cache-Speicher)
- Anwendung: CPU-Cache (L1, L2, L3...), Befehls- und Datencache, Festplattencache
- **CPU-Cache-Speicher**
  - kleiner schneller Pufferspeicher mit Kopien derjenigen Teile des Arbeitsspeichers, auf die am wahrscheinlichsten von der CPU als nächstes zugegriffen wird
  - entweder auf Prozessorchip integriert (on-chip-Cache) mit sehr kurzen registerähnlichen Zugriffszeiten oder prozessorextern (off-chip-Cache) als SRAM-Speicher
  - **Cachespeicherverwaltung** (sorgt dafür, dass am häufigsten benötigten Daten im Cache befinden)
  - **Hardware-Steuerung (Cache-Controller)** (kopiert alle Daten in Cache, auf die CPU zugreift, Auslagerung nach versch. Strategien)
  - Effizienz durch **Lokalitätseigenschaften** (siehe oben)
  - **Lesezugriffe**
    - **Cache-Hit** (Treffer bei Lesezugriff, Datum kann ohne Wartezyklen aus Cache entnommen werden)
    - **Cache-Miss** (kein Treffer, Datum mit Wartezyklen aus Arbeitsspeicher lesen und in Cache einfügen)
    - Hit-Rate (Trefferquote, = Anzahl Treffer / Anzahl Zugriffe)
    - mittlere Zugriffszeit (= Hit-Rate \* Zugriffszeit des Caches + (1 – Hit-Rate) \* Zugriffszeit ohne Cache)
  - **Schreibzugriffe**
    - bei Cache-Miss: Datum sowohl in Arbeitsspeicher als auch in Cache schreiben
    - bei Cache-Hit (also bestehendes Datum von CPU verändert) mehrere Verfahren
    - **Durchschreibeverfahren**
      - Datum von der CPU gleichzeitig in Cache und Arbeitsspeicher schreiben
      - Vorteil: **garantierte Konsistenz** zwischen Cache und Arbeitsspeicher
      - Nachteil: Schreibzugriffe benötigen langsame Zugriffszeit des RAM und belasten Systembus
    - **Gebuffertes Durchschreibeverfahren**
      - kleiner Schreib-Puffer, der zu schreibende Daten temporär aufnimmt, danach von Cache-Controller in Hauptspeicher übertragen, CPU arbeitet parallel weiter
    - **Rückschreibeverfahren**
      - Datum von CPU nur in Cache schreiben, durch ein spezielles Bit (altered Bit, modified Bit, dirty Bit) kennzeichnen, Arbeitsspeicher nur verändert, wenn so ein gekennzeichnetes Datum aus dem Cache verdrängt wird
      - Vorteil: Schreibzugriffe mit der schneller Cache-Zykluszeit
      - Nachteil: **Konsistenzprobleme** zwischen Cache- und Arbeitsspeicher
  - **Konsistenzprobleme**
    - andere Systemkomponenten (z.B. DMA-Controller) finden veraltete Daten im Arbeitsspeicher (von CPU geändert, aber noch nicht zurück geschrieben)
    - andere Systemkomponenten können Hauptspeicher verändern, während CPU noch mit alten Daten im Cache arbeitet (→ aufwendige Verfahren der Cache-Steuerung notwendig)
  - **Aufbau eines Cache-Speichers** (siehe Folie 6.147)
    - **Datenspeicher** (enthält im Cache abgelegte Daten)
    - **Adressspeicher** (enthält die Adressen dieser Daten im Arbeitsspeicher)
    - jeder Dateneintrag besteht aus ganzem Datenblock (bis 64 Byte), mit jedem zugegriffenen Datum wird Umgebung mit eingelagert (Hoffnung auf Lokalität), im Adressspeicher Basisadresse jedes Blocks
    - **Komparator** (ermittelt, ob das zu einer auf dem Adressbus liegende Adresse gehörende Datum auch im Cache abgelegt ist → Adressvergleich mit Adressen im Adressspeicher, muss schnell sein)
- **Datentransfer (DMA/PIO)**
  - **PIO** (Datenübertragung zwischen Speicher und Peripherie mittels Prozessor) (siehe Folie 6.159)
  - Nachteil: mindestens 4 Speicherzugriffe / Datum nötig → langsamer Datentransfer, Prozessor belastet
  - **Direkter Speicherzugriff (DMA)**
    - Datentransfer ohne Beteiligung der CPU direkt zwischen den beteiligten Komponenten
    - **DMA-Controller** (kontrolliert den Datentransfer)
    - Vorteil: Speicherzugriffe entfallen (nur 2 oder 1 Speicherzugriff / Datum nötig), Prozessor entlastet
    - Nachteil: **Cache-Inkonsistenzen** müssen ggf. behandelt werden
  - **Je nach Anwendung kann PIO oder DMA besser sein**
  - **Prinzip des direkten Speicherzugriffs** (siehe Folie 6.162)
    - CPU überträgt zunächst Startadresse, Zieladresse, Anzahl zu übertragender Bytes, Übertragungsrichtung, Steuerinformationen
    - „two cycle transfer“ – **Übertragungsverfahren** (Datum wird in einem Register des DMA-Controllers zwischengespeichert)
    - „fly by transfer“ – **Übertragungsverfahren** (Datum ohne Zwischenspeicherung direkt übertragen, gilt nicht für Speicher-Speicher Transfer)
  - **DMA-Übertragungsraten**
    - **Einzeltransfer** (jeweils genau ein Datum übertragen, danach Systembus wieder für CPU freigegeben)
    - cycle stealing, dem Mikroprozessor werden einzelne Buszyklen entzogen
    - **Block-Transfer** (überträgt alle Daten des Blocks ohne zwischenzeitliche Busfreigabe)
    - **Transfer auf Anforderung** (Datenübertragung solange ohne Unterbrechung solange REQ-Signal aktiv)
    - burst mode, Datentransfer in Schüben, häufig von Peripheriegeräten benutzt

- oftmals **unterschiedliche Datenbreiten** (z.B. von 8-Bit Geräten (Drucker) in 32-Bit Hauptspeicher)
- oftmals **Übertragung von non-aligned Daten** (mehrere Speicherzugriffe zum Lesen oder Schreiben eines Datums notwendig)
- **DMA-Controller wahlweise angepasst**
- **Systemsteuerbausteine**
  - übernehmen Funktionen, die aus technologischen oder Kostengründen nicht im Prozessor integriert sind
  - **Nicht programmierbare Systemsteuerbausteine**
    - führen fest vorgegebene, vom Prozessor nicht beeinflussbare Funktion aus
    - z.B. Taktgenerator, Bus-Controller, Bus-Arbitrer, DRAM-Controller
  - **Programmierbare Systemsteuerbausteine**
    - Funktionsweise durch Steuerworte vom Prozessor beeinflussbar
    - z.B. DMA-Controller, Cache Controller, MMU, Timer, Real-Time Clocks, Interrupt Controller
    - programmierbare Systembausteine erscheint für CPU wie ein kleiner Satz von Adressen (Portadressen)
    - **Speicherbezogene Adressierung (speicherbezogene Ein-/Ausgabe, Memory Mapped IO)**
      - Adressblock in gemeinsamen Adressraum mit allen anderen Speicheradressen untergebracht
      - kein Unterschied zwischen Speicheradresse und Adresse eines Registers eines Peripherie-Bausteins
      - häufig zusammenhängender Speicherbereich für Peripherie-Bausteine verwendet (I/O-Page)
    - **isolierte Adressierung (isolierte Ein-/Ausgabe, Isolated IO)**
      - zwei getrennte Adressräume für Speicher und Ein-/Ausgabe (eigener I/O-Adressraum)
      - Auswahl des Adressraums durch zusätzliche Signal M/IO (memory / input-output)
  - **Programmierbare Interrupt-Controller (PIC)**
    - Spezialbausteine, die mehrere Interrupt-Controller verwalten und koordinieren, entlasten CPU
    - Interrupt Quellen melden über Leitungen Unterbrechungswünsche an den Controller
    - Controller wählt Quelle nach Priorität und meldet diese an CPU weiter, wertet Interrupt aus und überträgt den entsprechenden Interrupt-Vektor an CPU
- **Schnittstellenbausteine (I/O-Controller)**
  - dienen als Bindeglied zwischen Prozessor / Arbeitsspeicher und der Peripherie
  - Aufgaben: Pufferung von Ein-/Ausgabe-Daten, Anpassung unterschiedlicher Arbeitsgeschwindigkeiten, Umsetzung der Daten (seriell-parallel, digital-analog), Erzeugung von Steuersignalen für Peripheriegeräte, Annahme und Erzeugung von Unterbrechungsanforderungen für Peripheriegeräte
  - **Bausteine für parallele Schnittstellen**
    - Übertragung von Daten zwischen Prozessor und Peripheriegeräten in paralleler Form
    - Peripheral Interface Adapter (PI, PIA)
    - Programmable Peripheral Interface (PPI)
    - Parallel I/O Circuit (PIO)
  - **Bausteine für serielle Schnittstellen**
    - Übertragung von Daten zwischen Prozessor und Peripheriegeräten in serieller Form
    - geringer Leitungsaufwand, große überbrückbare Entfernungen
    - Problem: Synchronisation zwischen Sender und Empfänger
      - Synchroner Übertragung (öffentliche Datennetze, lokale Netzwerke)
      - Asynchroner Übertragung (zwischen CPU und langsamen Peripheriegeräten)
- **Festplatten, RAID, CD, DVD, CRT, LCD, Modem, Printer, etc** (siehe Folien 6.183ff und Vorträge)
- **Embedded Systems (Car, Communication Technology, Robotic Systems, Web server for Industry Control Applications, Sensor Boards for Sensor Nets), Scatterweb, Sensor Networks, Outlook** (siehe Folien 7.1ff)

### Wichtige Abbildungen im Skript

- Digitalrechner nach dem von-Neumann Konzept (Folie 1.12)
- **Interner Aufbau eines einfachen Mikroprozessors (Folie 2.3)**
- Mic-1 (Folie 2.95)
- Strukturelle Maßnahmen zur Leistungssteigerung in Rechnersystemen (Folie 2.103ff)
- **Pipelining (Folie 2.116ff)**
- ISA-Level als Schnittstelle zwischen Hardware und Software (Folie 3.3)
- Registerfenster, Befehlsformate, Adressierungsarten (Folie 3.31ff)
- Programmablauf, Prozeduren, Co-Routinen (Folie 3.88ff)
- Polling (Folie 3.104f)
- **Daisy-Chain-Verfahren (Folie 3.108)**
- **Betriebssystem verkleinert semantische Lücke zwischen Anwendung und Hardware (Folie 4.4)**
- Virtuelle und Reelle Adressen, Segmentierung (Folie 4.15ff)
- Virtueller und Physikalischer Cache (Folie 4.38f)
- **Architektur eines klassischen PC-Systems (Folie 6.6)**
- Klassifizierung von Halbleiterspeicher (Folie 6.79)
- Big Endian und Little Endian (Folie 6.110)
- **Speicherhierarchie (Folie 4.35, Folie 6.125f, Folie 6.133)**
- Cache (Folie 6.137ff)
- DMA/PIO (Folie 6.159, Folie 6.162)

**Begriffserläuterungen (\* mehrdeutig)**

CPU = Central Processing Unit  
 CU = Control Unit  
 AU = Address Unit  
 ALU = Arithmetic Logical Unit  
 BIU = Bus Interface Unit  
 PC = Program Counter  
 IP = Instruction Pointer  
 SP = Stack Pointer  
 CCR = Condition Code Register  
 SISD = Single Instruction Single Data  
 SIMD = Single Instruction Multiple Data  
 MIMD = Multiple Instruction Multiple Data  
 MISD = Multiple Instruction Single Data  
 CPI, \*IPC = Cycles per Instruction / Instruction per Cycle

IRQ, INT, INTR = Interrupt Request  
 NMI = Non Maskable Interrupt  
 HALT = Interruptanforderung zum Stoppen des Prozessors  
 ISR = Interrupt Service Routine  
 IVN = Interrupt-Vektor-Nummer  
 PSW = Program Status Word / Prozessorstatuswort

BUS = Binary Unit System  
 HOLD, BR, BREQ = Bus Request  
 HOLDA, BG, BGRT = Bus Grant / Hold Acknowledge  
 \*BGA = Bus Grant Acknowledge  
 BERR = Bus Error  
 DBP = Datenbuspuffer  
 AP = Adresspuffer  
 TDM = Time Division Multiplexing (Zeitmultiplex)

\*ISA = Instruction Set Architecture  
 CISC = Complex Instruction Set Computer  
 RISC = Reduced Instruction Set Computer  
 \*IPC = Inter Process Communication  
 MMU = Memory Management Unit  
 API = Application Programming Interface  
 \*IDE = Integrated Development Environment  
 DLL = Dynamic Link Library

DIP = Dual Inline Packages  
 LCC = Leadless Carrier Chip  
 SMD = Surface Mounted Device  
 PGA = Pin Grid Array  
 \*BGA = Ball Grid Array  
 TTL-Pegel = Transistor Transistor Logic  
 \*ISA = Industry Standard Architecture (ISA-Slot)  
 PCI = Peripheral Component Interconnect  
 SCSI = Small Computer System Interface  
 \*IDE = Integrated Drive Electronics

AS = Address Strobe  
 DTACK = Data Transfer Acknowledge  
 ROM = Read-Only Memory  
 PROM = Programmable Read-Only Memory  
 EPROM = Erasable Programmable Read-Only Memory  
 EEPROM = Electronically Erasable Programmable Read-Only Memory  
 SRAM = Static Random Access Memory  
 DRAM = Dynamic Random Access Memory  
 RAS = Row Address Strobe  
 CAS = Column Address Strobe  
 FPM-DRAM = Fast Page Mode DRAM  
 EDO-RAM = Extended Data Output RAM  
 SDRAM = Synchronous Dynamic RAM  
 DDR-SDRAM = Double-Data-Rate-SDRAM  
 SLDRAM = Sync Link RDRAM  
 RDRAM = Rambus Dynamic RAM  
 SIPP = Single Inline Pin Package  
 SIMM = Single Inline Memory Module  
 DIMM = Dual Inline Memory Module  
 RIMM = Rambus Inline Memory Module

\*PIO = Peripheral Input/Output / Programmed Input/Output  
 DMA = Direct Memory Access  
 PIC = Programmierbare Interrupt Controller  
 PI, PIA = Peripheral Interface Adapter  
 PPI = Programmable Peripheral Interface  
 \*PIO = Peripheral I/O Circuit