Betriebssysteme

Vorlesung im Studiengang Bsc. Wirtschaftsinformatik

Dipl.-Wirt.Inform. Mike Heuser

Modulziele

- Mit erfolgreichem Abschluss dieses Moduls können die Studierenden
  - Betriebssysteme als Software-Fundament für IT-Architekturen auswählen und beurteilen,
  - Konzepte des Systembetriebs, von Batch-Systemen über Multitasking-Varianten, virtuellen Systemen bis hin zur Echtzeit-fähigkeit, erklären und beurteilen,
  - wichtige Architekturvarianten von Betriebssystemen beschreiben und existierende Betriebssysteme diesen Varianten zuordnen,
  - grundlegenden Mechanismen zur Verwaltung und Abstraktion der Betriebsmittel (Prozess, Thread, virtueller Speicher, Dateisystem, Kommunikation und Synchronisation) verstehen und durch praktische Anwendung nutzen,
  - die Interaktion Betriebssystem / Hardware mit Hilfe von Ausnahmebehandlungen erklären und beurteilen,
  - einen Überblick über wesentliche Bestandteile wie Oberflächen, Werkzeuge und Dateisysteme geben.

Prüfungen

1. Klausur über 90 Minuten
   - Davon ca. 10% Transferaufgabe
   - 6 ECTS-Punkte

Literaturempfehlung

- **Moderne Betriebssysteme**
  - Tanenbaum, Andrew S., Bos, Herbert
  - 4., aktualisierte Auflage, Pearson Studium, 2016, ISBN: 978-3868942705

- **Rechnerarchitektur. Von der digitalen Logik zum Parallelrechner**
  - Tanenbaum, Andrew S; Austin,Todd
  - 6., aktualisierte Auflage 2014, Pearson Studium, ISBN: 978-3868942385

EINFÜHRUNG

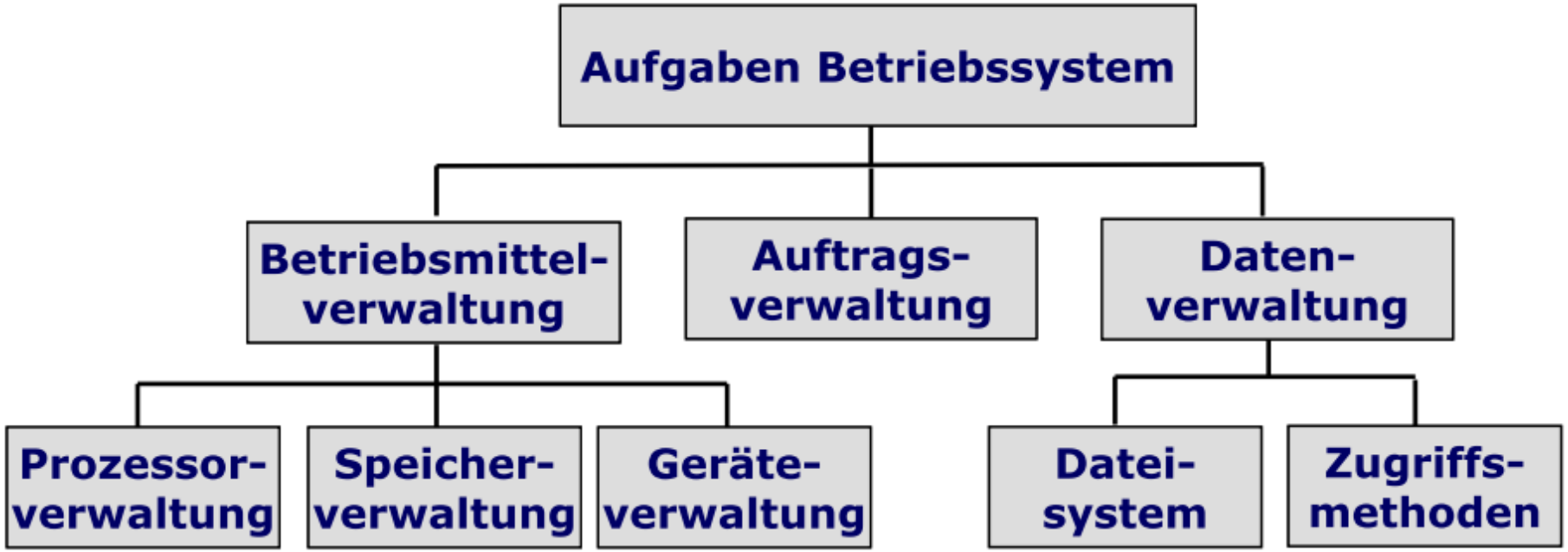
Gliederung

- Einführung
- Betriebssystemstrukturen
- Betriebsarten
- Betriebssystem und Hardware

Definition Betriebssystem

- Gabler Wirtschaftslexikon:
  Sammelbegriff für Programme (Systemprogramme), die den Betrieb eines Computers erst möglich machen, auch als *Operating System (OS)* bezeichnet. Sie **steuern** und ü**berwachen** das Zusammenspiel der Hardwarekomponenten im Rahmen der **Auftrags-, Daten-, Arbeitsspeicher- und Programmverwaltung** (besonders die Abwicklung einzelner Anwendungsprogramme, den Zugriff von Prozessen auf bestimmte Ressourcen) sowie der Systemsicherung (Fehlererkennung und -behebung).
  Das Betriebssystem macht ein Datenverarbeitungssystem erst bedienbar und beherrschbar.

Aufgaben


Aufgaben Betriebssystem
Betriebsmittel-verwaltung
Auftrags-verwaltung
Daten-verwaltung
Prozessor-verwaltung
Speicher-verwaltung
Geräte-verwaltung
Datei-system
Zugriffs-methoden

Historie

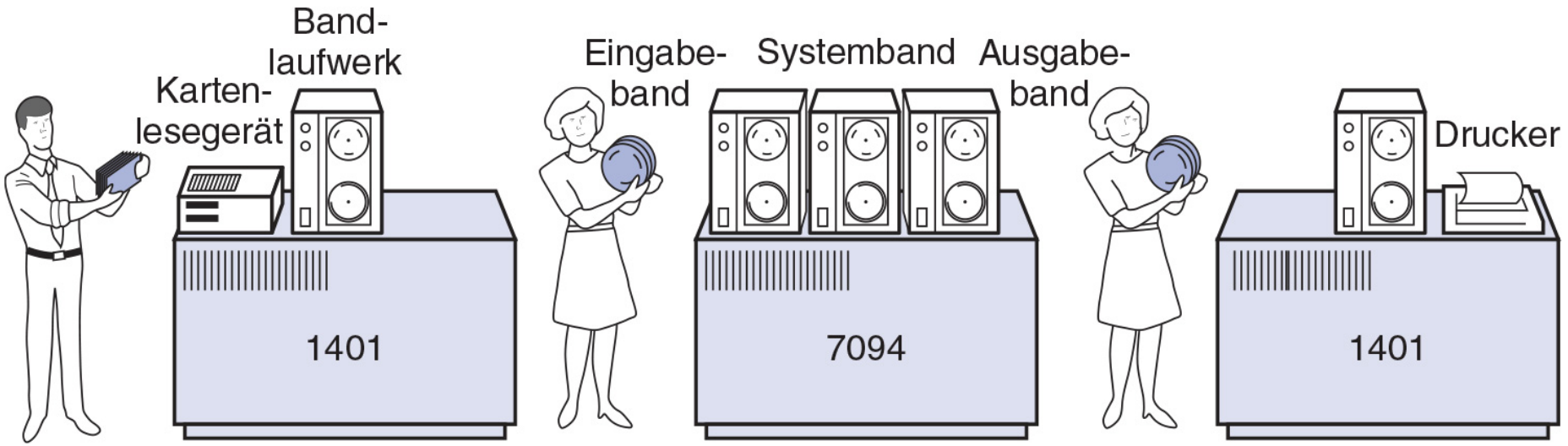
- Vorzeit"
  - Analytical Engine, Charles Babbage (1792 - 1871)
  - Erster Digitalrechner, programmierbar
- Die erste Generation (1945 - 1955)
  - Wichtige Personen
    - Konrad Zuse, Deutschland
    - Howard Aiken, Harvard University
    - J. Presper Eckert / William Mauchley, University of Pennsylvania
    - John v. Neumann, Princeton
  - Maschinensprache, Festverdrahtet
  - Relais, Röhren
  - Steckkarten, Lochkarten
  - Einfache mathematische Berechnungen
    - Tabellen mit Sinus- / Cosinuswerten

Historie

Die zweite Generation (1955 - 1965)

- Mainframes
- Transistoren
- Lochkarten in Assembler / Fortran
  - Job Abarbeitung von Lochkartenstapeln durch Operator
- Vermarktungsfähig
  - Entstehung der Personengruppen Entwickler / Hersteller / Operator / Programmierer / Wartungspersonal
- Stapelverarbeitungssysteme
  - Kleinrechner (IBM 1401) liest Lochkarten auf Magnetbänder ein
  - Großrechner (IBM 7094) verarbeitet Daten und führt Berechnungen aus
  - Datenausgabe auf Drucker erneut auf kleinem Rechner
  - Bedienung durch Operator

Historie


Band-
laufwerk
Karten-
lesegerät
Eingabe-
band
Systemband
Ausgabe-
band
Drucker
1401
7094
1401

Historie

Die dritte Generation (1965 - 1980)

- IBM System/360
  - Vereinigung von unterschiedlichen Produktlinien
    - Zeichenorientierte Geräte (IBM 1401)
    - Wortorientierte Geräte (IBM 7094)
  - Gleiche Architektur und Befehlssatz bei verschiedenen Leistungsstufen
  - Erstmals Verwendung von ICs
- OS/360 Betriebssystem
  - Fehlerbehaftung durch sehr hohe Komplexität des Systems
    - Unterschiedlichste Peripheriegeräte
    - Wissenschaftliche / Kommerzielle Anwendungen
  - Multiprogramming
    - Eigene Speicherpartitionen für mehrere Jobs
    - CPU wechselt zwischen den Jobs bei Wartezeiten

Historie

Die dritte Generation (1965 - 1980)

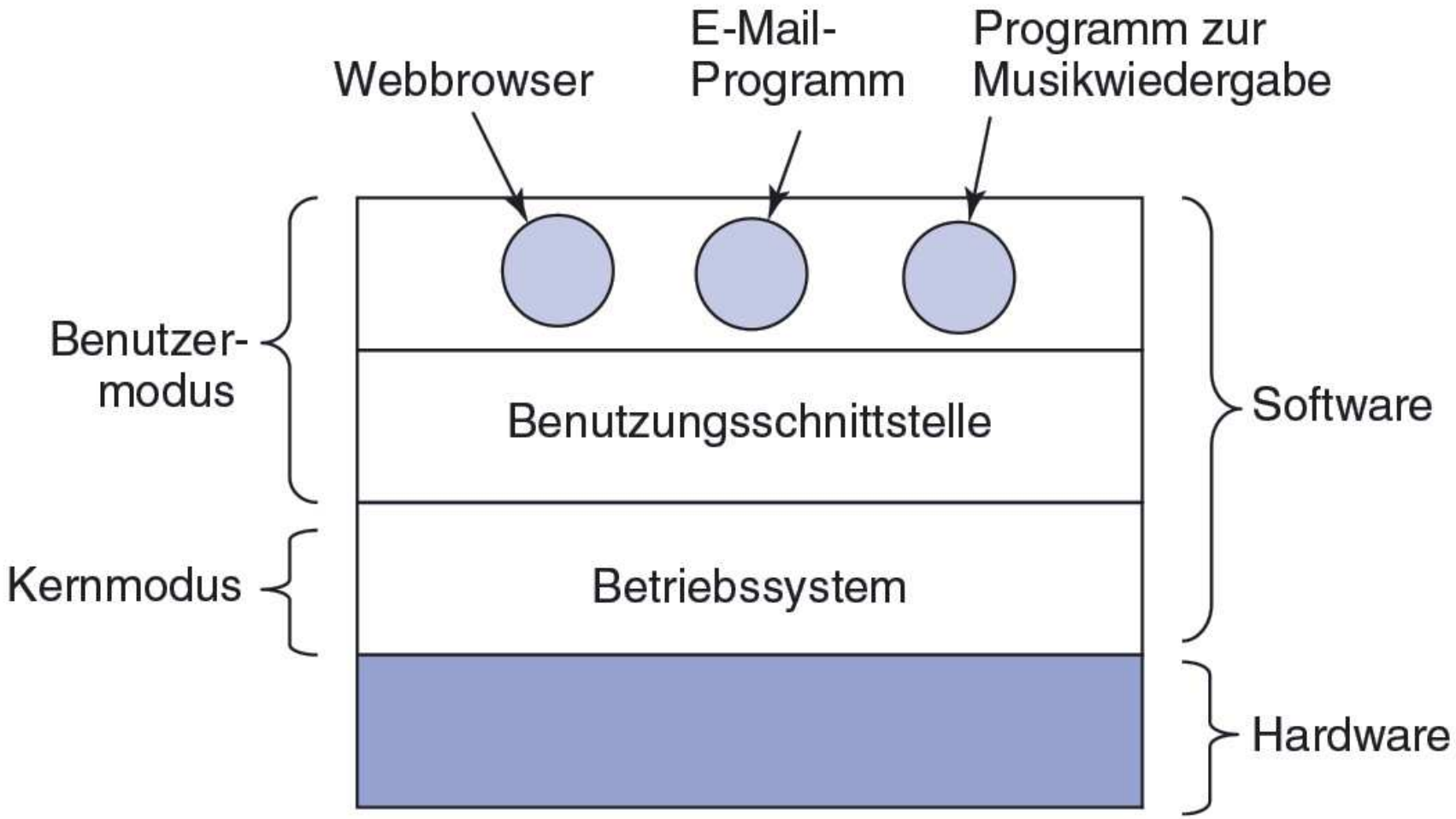
- OS/360 Betriebssystem
  - Spooling
    - Jobs werden von Lochkarten eingelesen und auf Festplatte zwischengespeichert
    - Bearbeitung unmittelbar nach Beendigung eines abgeschlossenen Jobs
  - Timesharing
    - Quasiparalleler Terminalzugriff durch mehrere Benutzer
    - Nutzung der Rechnerzeit nur durch aktive Jobs / Nutzer
    - Compatible-time-Sharing-System (CTSS), MIT
- MULTICS
  - MULTIplexed Information and Computer System
  - Mehrere hundert Benutzer
  - Rechenleistung entspricht späterer i80386 CPU

Historie

Die vierte Generation (1980 bis heute)

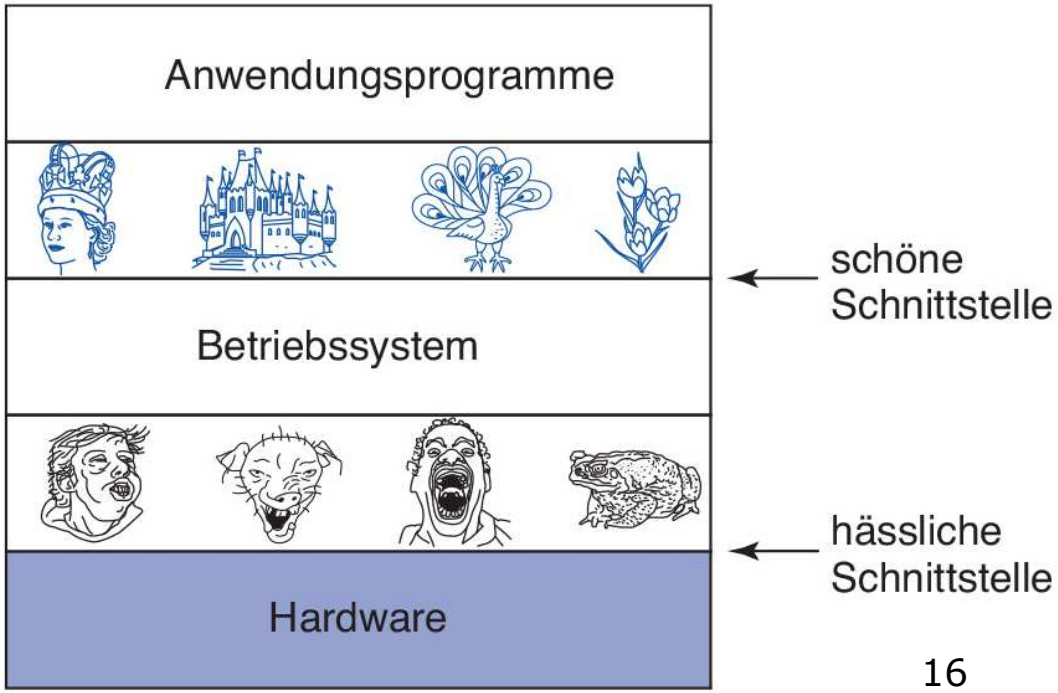
- Microcomputer
  - LSI-Schaltkreise
- Altair 8800
  - Intel 8080
  - Altair Basic
- CP/M
  - Gary Kildall
- QDOS
  - Seattle Computer Products
- Apple I, Apple II
- IBM PC
  - MS-DOS

Einordnung des Betriebssystems



Aufgaben

- Abstraktion der Betriebsmittel für Anwendungsprogrammierer /-programmen, nicht für Benutzer
- Verwaltung der Hardware / Ressourcen
  - zeitlich: z.B. CPU
  - räumlich: z.B. Speicher



Verwaltung Beispiele

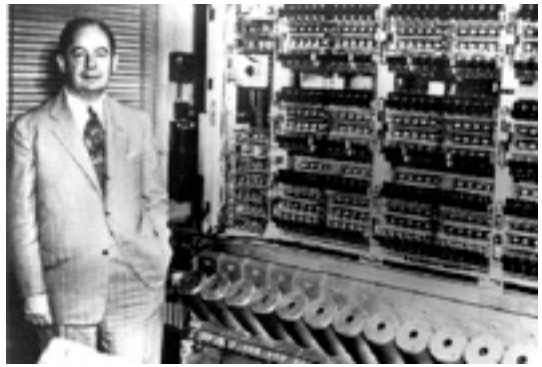
- Drucker
  - 2 User wollen gleichzeitig auf Arbeitsgruppendrucker drucken
  - OS regelt den Zugriff
    - User A wird in einer Warteschlange hinter User B einsortiert und die Schlange sequentiell abgearbeitet.
- Dateien:
  - Jeder Datei werden Rechte (Benutzer/Gruppen) zugeordnet
    - lesen/schreiben/löschen
  - Einhaltung wird durch OS sichergestellt
    - Z.B. wird unbefugter Zugriff verweigert
- Anwendung können nicht auf die Hardware direkt zugreifen (unkontrolliert!)

Aufbau von Computersystemen

- Prozessoren
- Hauptspeicher
- Sekundärspeicher
- Eingabe/Ausgabe

John von Neumann

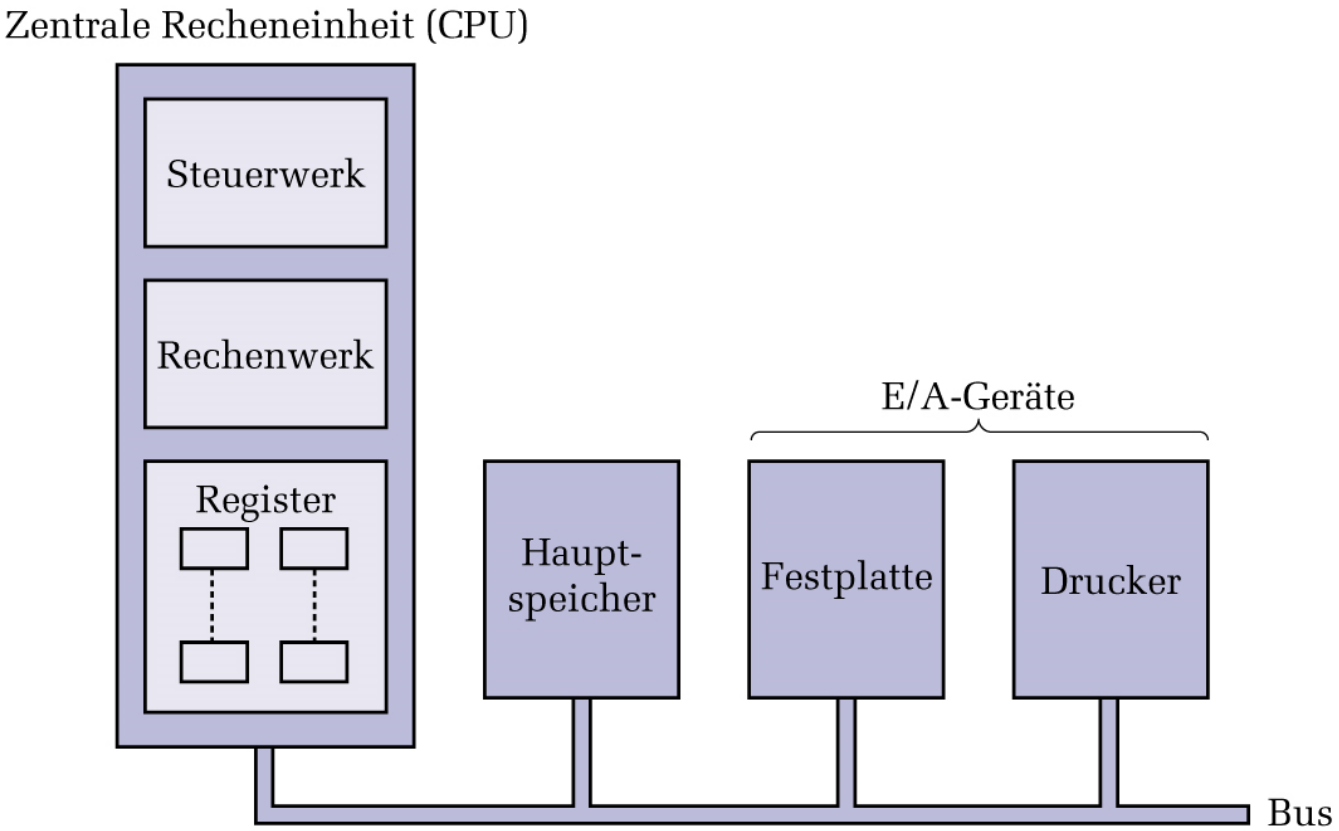
- US-amerikanischer Mathematiker ungarischer Herkunft,
  * 28. 12. 1903 Budapest, † 8. 2. 1957 Washington D.C.
- Lehrte in Princeton, N.J.; arbeitete über Gruppen- und Funktionentheorie sowie über elektronische Rechenanlagen



Von-Neumann-Architektur

- Architekturkonzept für speicherprogrammierten Universalrechner (1944)
  - sog. „von Neumann Architektur“
  - Nicht-“von Neumann“ Rechner blieben bislang ohne größere Bedeutung
- Die Von-Neumann Architektur wird bis heute in den meisten Computersystemen realisiert

Von-Neumann Rechner
Komponenten



Von-Neumann Rechner Konzept

- 1. Merkmal: Ein programmgesteuerter Rechner besteht aus den folgenden Komponenten
  - Rechenwerk (ALU)
  - Steuerwerk (CU)
  - Speicherwerk / Datenspeicher
  - Eingabewerk / Ausgabewerk
  - interne Datenwege (Busse)

Von-Neumann Rechner Konzept

- 2. Merkmal: Der Rechner verwendet eine binäre Zahlendarstellung (Dualzahlen).
- 3. Merkmal: Die Struktur des Rechners ist unabhängig von dem zu bearbeitenden Problem, die Anpassung an die unterschiedlichen Aufgaben wird mittels Software gelöst.
- 4. Merkmal: Programme und Daten werden in einem gemeinsamen Speicher abgelegt. Alle Speicherplätze sind gleich lang und werden über Adressen einzeln angesprochen.
- 5. Merkmal: Die Befehle geben nur die Speicheradresse an, wo die Daten abgelegt sind, nicht die Daten selbst.

Prozessoren

Zentralprozessor

- Central Processing Unit (CPU)
  - Zusammenspiel des Steuerwerks mit dem Rechenwerk sorgt für korrekten Programmablauf
  - Steuerwerk ruft Befehle aus dem Hauptspeicher ab und bestimmt Befehlstyp
  - Rechenwerk ist für arithmetische Operationen zuständig, wie z. B. Addition oder logische Operationen, wie AND und NOT
  - Rechenwerk hat eine feste Verarbeitungsbreite, z. B. von 8, 16, 32 oder 64 Bit. Man spricht dann von einem 8, 16, 32 oder 64 Bit-System

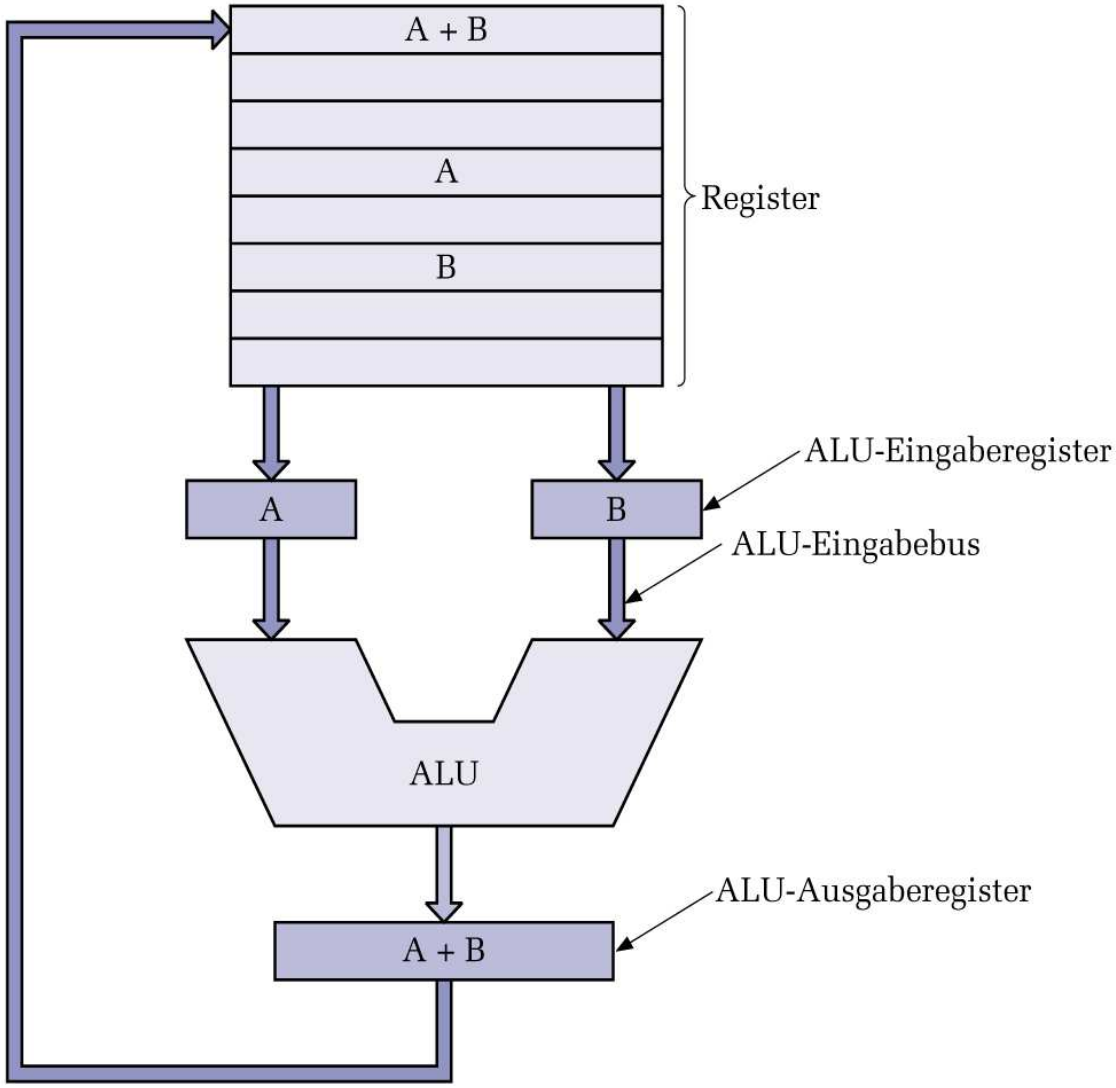
Steuerwerk

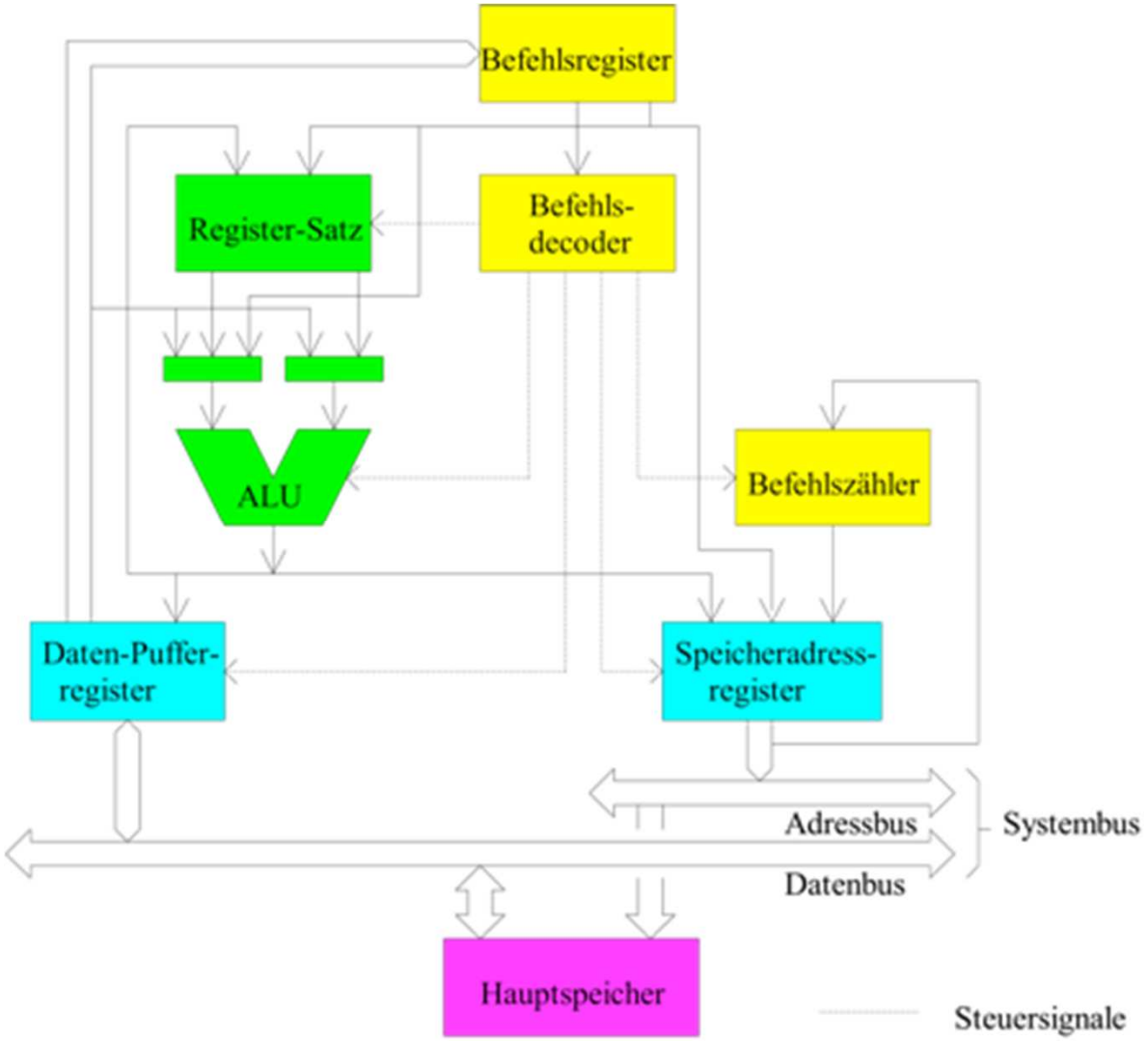
- Control Unit (CU)
  - Koordination der zeitlichen Abläufe im Rechner
- Befehlszähler (Program Counter – PC)
  - zeigt auf den nächsten Befehl
- Befehlsdecoder
  - Entschlüsselt und verarbeitet den Befehl und erzeugt die notwendigen Hardware-Steuersignale
- Befehlsregister (Instruction Register - IR)
  - Speichert den aktuell zu bearbeitenden Befehl
- Speicheradressregister (Memory Address Register)
  - Adresse des nächsten Befehls oder eines Datenwortes

Rechenwerk

- Arithmetisch-logische Einheit / Arithmetic Logical Unit (ALU)
  - Verknüpfung der beiden Eingänge A und B
  - Rein kombinatorisch aufgebaut, braucht keinen Takt
  - Befehl bestimmt die Operation
  - CPU-Hardware wandelt Befehlscode in Steuersignale um

Von-Neumann-Rechner
Datenpfad




Befehlsregister
Register-Satz
Befehls-
decoder
ALU
Befehlszähler
Daten-Puffer-
register
Speicheradress-
register
Adressbus
Systembus
Datenbus
Hauptspeicher
Steuersignale

RISC & CISC Architektur

- RISC
  - Reduced Instruction Set Computer
- CISC
  - Complex Instruction Set Computer

CISC Architektur

- Effektivität der Programmierung wird durch komplexe Befehle erhöht
- Reduktion des Programmcodes durch in die CPU integrierten Microcode
- Beispiel:
  - MOVEM.L D5-D8,(A4)
  - führt folgende Befehle aus:
    - MOVE.L D5,(A4)
    - MOVE.L D6,(A4)
    - MOVE.L D7,(A4)
    - MOVE.L D8,(A4)

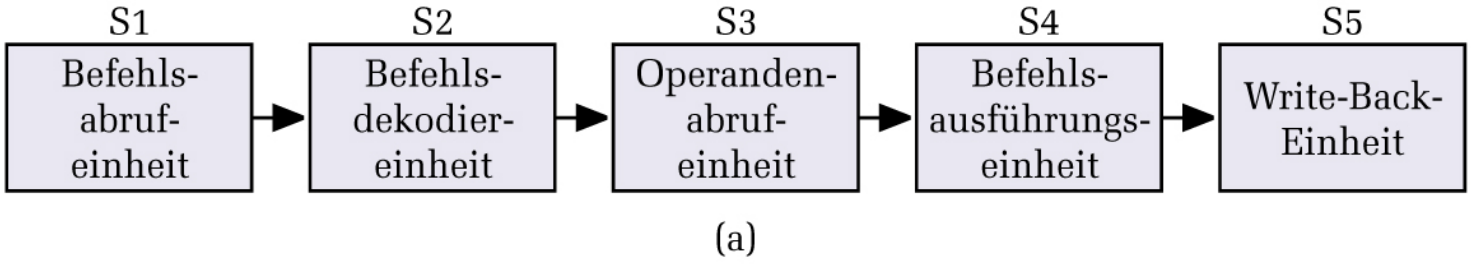
Eigenschaften von RISC

- Reduzierter Befehlssatz
- Nur festverdrahtete Mikrobefehle
  - keine auf der CPU gespeicherten Mikroprogramme
- Erweiterte RISC Technologien
  - Erhöhte Registeranzahl
  - Sprungvorhersage im Compiler
  - zumeist einheitliche Befehlslänge
    - CISC Befehle variieren in der Befehlslänge

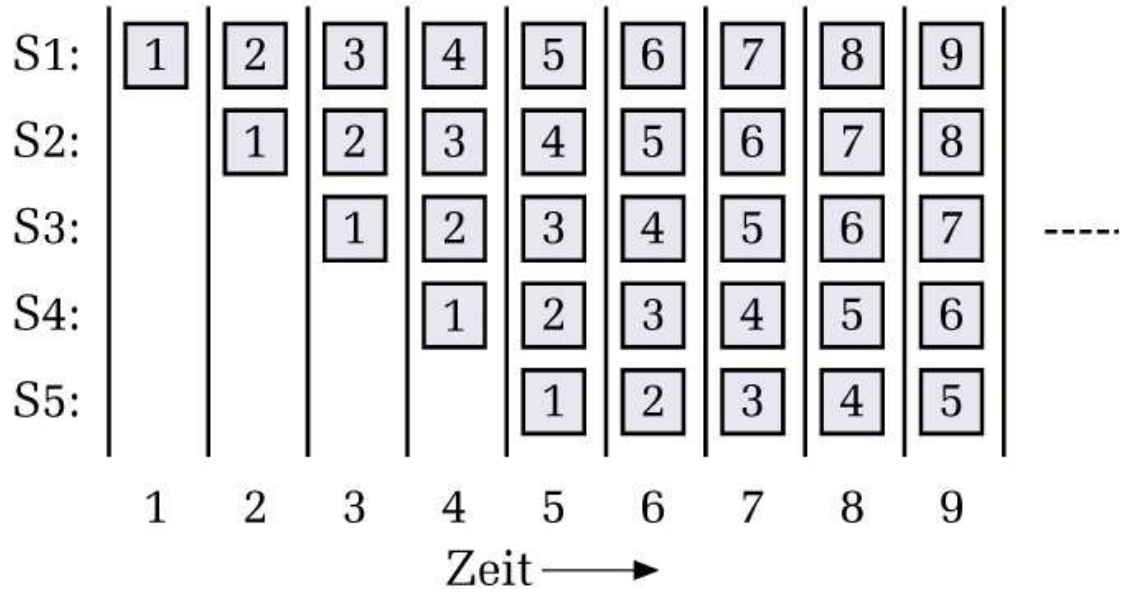
Von-Neumann-Zyklus

- Aufteilung in 5-stufige Befehlsverarbeitung
  - Befehl holen - fetch
  - Befehl decodieren - decode
  - Operanden holen – fetch operands
  - Befehl ausführen - execute
  - Ergebnis speichern – write back

Parallelität auf Befehlsebene
Pipelining

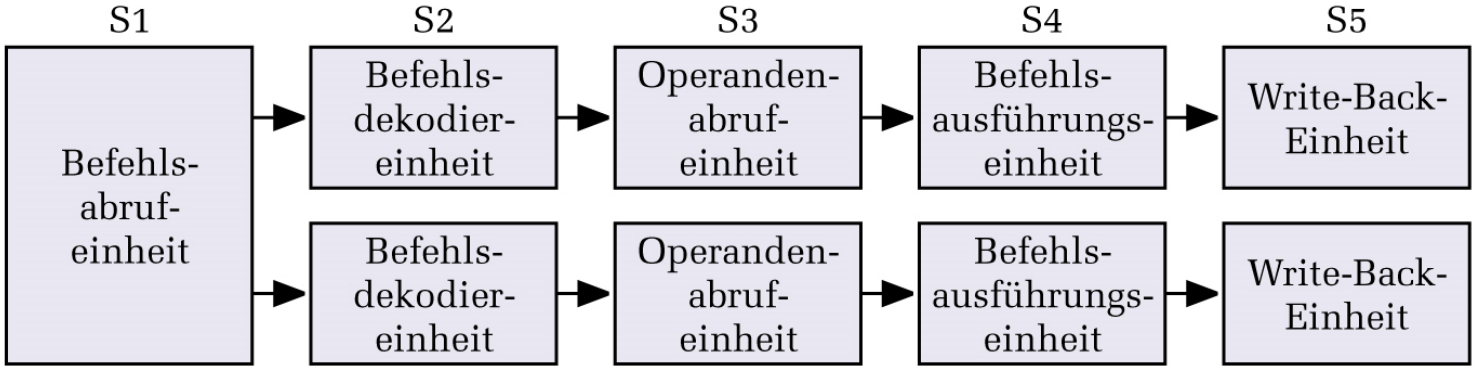


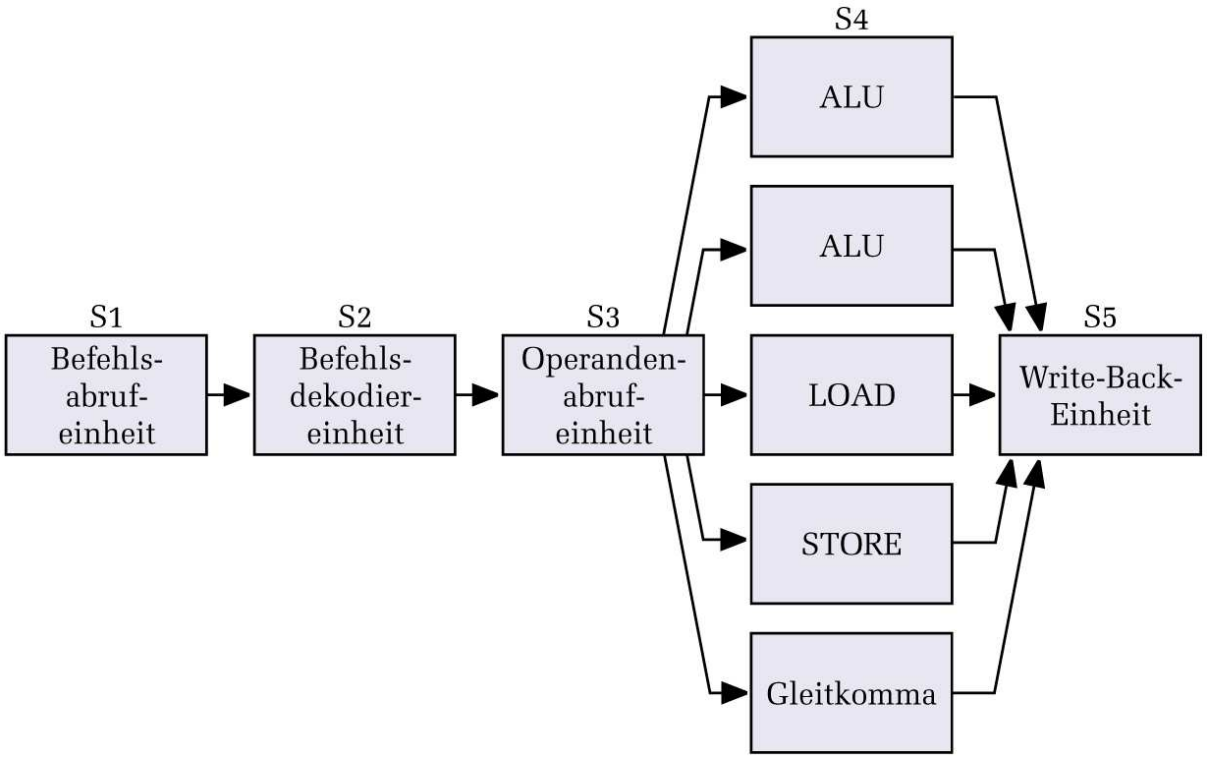
(a)



(b)

Superskalare Architektur

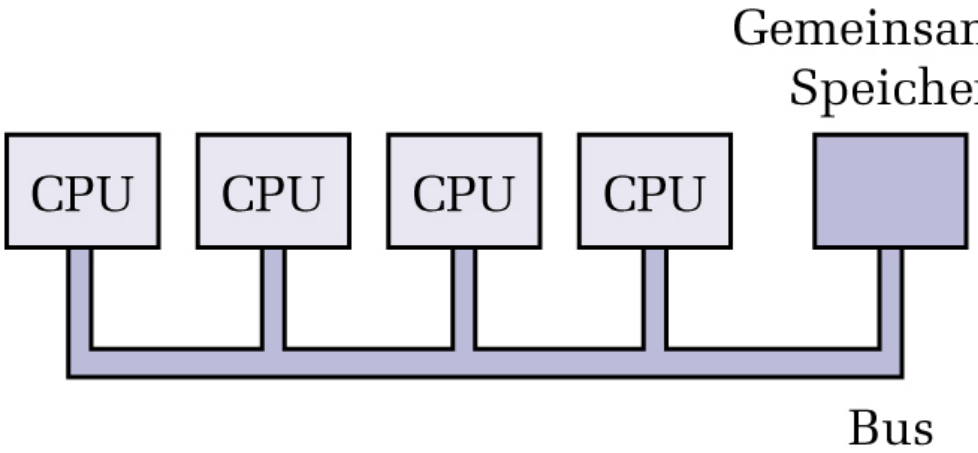




Parallelität auf Prozessorebene
Mehrprozessorsysteme

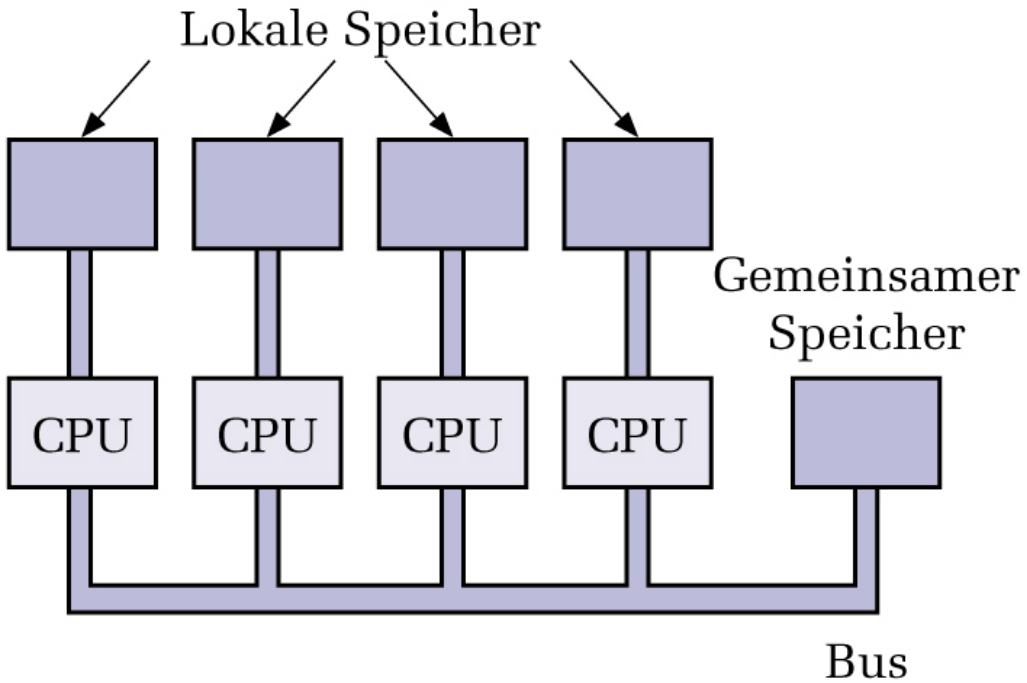
- Zwei oder mehr Prozessoren
- Höhere Rechenleistung
- Gliederung in
  - Symmetrische Multiprozessorysteme (SMP)
    - Gemeinsamer Adressraum für alle identischen Prozessoren
    - gleichrangiger Zugriff auf E/A-Geräte und Speicher
    - Dynamische Verteilung der Prozesse auf die CPUs
  - Asymmetrische Multiprozessorsysteme (AMP)
    - Ein Prozessor verteilt Aufgaben an die weiteren Prozessoren
    - Traditionell identische Prozessoren, aber auch Unterteilung in Haupt- und nachrangige Prozessoren möglich
    - Auch die Nutzung von GPU(s) und Physics-Prozessoren durch die CPU entsprechen i.W.S. dem Konzept des AMP

Parallelität auf Prozessorebene
Mehrprozessorsysteme



eng gekoppelte
Multiprozessorsysteme

lose gekoppelte
Multiprozessorsysteme



Busse

- Gemeinsam genutzter Weg, an den mehrere Funktionseinheiten angeschlossen werden können.
- Zu jedem beliebigen Zeitpunkt kann nur eine der angeschlossenen Einheiten den Bus zur Übertragung nutzen.
- Der Bus besteht aus
  - Adress-
  - Daten- oder
  - Steuerleitungen

Aufgaben

1. Nennen Sie vier Betriebsmittel, welche das Betriebssystem verwaltet! Welche davon sind hardware- und welche softwaretechnische Betriebsmittel?
2. Wozu verwendet man Embedded Systems? Nennen Sie ein Beispiel!

PROZESSE

Prozesse

- Multiprogrammiersystem
  - CPU wechselt von Programm zu Programm (10-100 ms)
  - Nur exakt ein Programm zur selben Zeit
    -> Quasiparallelität
- Multiprozessorsysteme
  - Zwei oder mehr CPUs teilen sich einen gemeinsamen Speicher

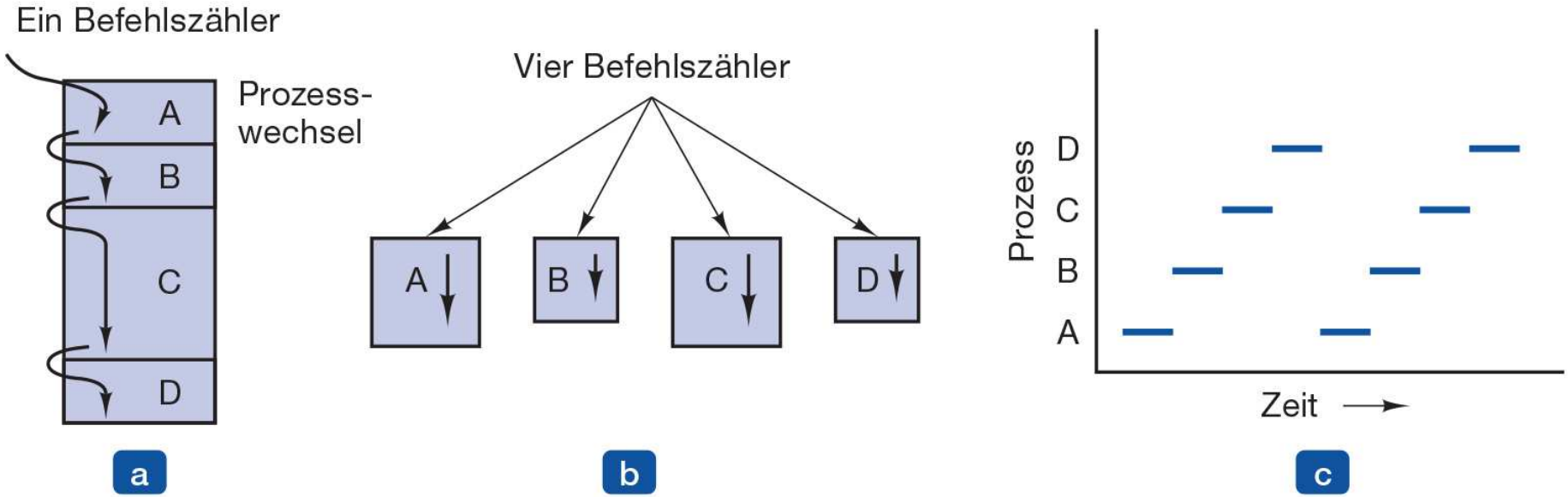
Prozessmodell

- Auf dem Computer ausführbare Software ist als Menge von sequentiellen Prozessen (kurz: Prozesse) organisiert.
- Prozess ist eine Instanz eines **Programms in Ausführung** auf einer eigenen virtuellen CPU (konzeptionell) mit
  - Identifikator (Prozess-ID)
  - Prozesszustand
  - aktuellem Wert des Befehlszählers,
  - den Registerinhalten,
  - Zugriffsrechte,
  - aktuell belegte Betriebsmittel,
  - der Belegung der Variablen
  - ….

  => Prozesskontrollblock (PCB)

Prozesstabelle
Prozesskontrollblock PCB

Prozessverwaltung	Speicherverwaltung	Dateiverwaltung
Register	Zeiger auf Textsegment	Wurzelverzeichnis
Befehlszähler	Zeiger auf Datensegment	Arbeitsverzeichnis
Programmstatuswort	Zeiger auf Stacksegment	Dateideskriptor
Stackpointer		Benutzer-ID
Prozesszustand		Gruppen-ID
Priorität		
Scheduling-Parameter		
Prozess-ID		
Elternprozess		
Prozessgruppe		
Signale		
Startzeit des Prozesses		
Benutzte CPU-Zeit		
CPU-Zeit der Kindprozesse		
Zeitpunkt des nächsten Alarms		



(a) Multiprogrammierung von vier Programmen. (b) Konzeptionelles Modell von vier individuellen sequenziellen Prozessen. (c) Zu jedem Zeitpunkt ist immer nur ein Programm aktiv.

Programm vs. Prozess

- Prozess
  - Ist eine **Aktivität** jedweder Art,
  - umfasst Programm, Eingabe, Ausgabe und einen Zustand.

- Programm
  - Wird auf einem Datenträger gespeichert und macht nichts.

- Je Programm wird mindestens ein Prozess gestartet.
- Wird das selbe Programm ein weiteres mal gestartet, so werden auch hier zwei Prozesse ausgeführt.
  - Evtl. kann der Code zwischen beiden aufgeteilt werden, so dass nur eine Kopie im Arbeitsspeicher ist.

Prozesserzeugung

Einfache Systeme

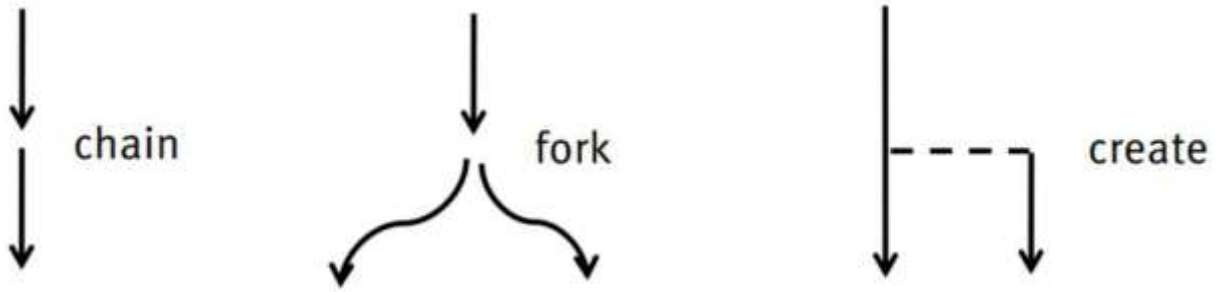
- häufig nur ein Prozess, der beim Start erzeugt wird und auf Benutzeranfragen reagiert
- Embedded System / eingebettete Systeme (Mikrowelle, Waschmaschine, TV…)

Komplexere Systeme

- Ereignisse für Prozessstart
  - Initialisierung des Systems
  - Systemaufruf zum Erzeugen eines Prozesses durch einen anderen Prozess
  - Benutzeranfrage, einen neuen Prozess zu erzeugen
    - „Doppelklick"
  - Initiierung einer Stapelverarbeitung
    - Batchjobs/Stapeljobs (Großrechner)
- Prozesse im Hintergrund heißen Daemons

Erzeugung von Prozessen

- **Prozessverkettung (chaining):** der laufende Prozess startet einen neuen Prozess und terminiert sich damit selbst
- **Prozessvergabelung (forking):** Der laufende Prozess startet einen neuen Prozess, läuft selbst aber weiter.
- **Prozesserzeugung (creation):** Der laufende Prozess startet einen unabhängigen neuen Prozess (Variante der Prozessvergabelung)



Prozessbeendigung

Einfache Systeme

- Beendigung des Prozess durch Abschalten des Gerätes

Komplexe Systeme

- Normales Beenden (freiwillig)
  - Programm beenden, „X“
- Beenden aufgrund eines Fehlers (freiwillig)
  - Bspw. Datei nicht gefunden
- Beenden aufgrund eines schwerwiegenden Fehlers (unfreiwillig)
  - Programmierfehler (Div/0, ungültige Speicheradresse…)
- Beenden durch einen anderen Prozess (unfreiwillig)
  - „Kill“

Prozesshierarchie

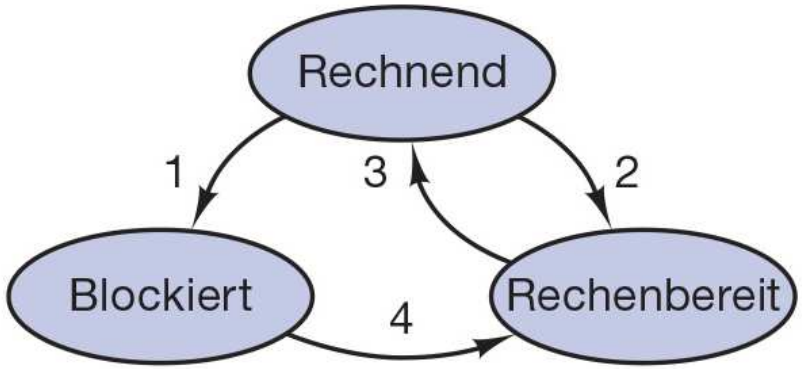
- Wird ein neuer Prozess erzeugt
  - Elternprozess - Kindprozess
  - Kindprozess weitere Prozesse -> Prozesshierarchie
- Unix
  - Prozess mit allen Kindern (Enkel…) -> Prozessfamilie
- Windows
  - kein Konzept einer Prozesshierarchie
  - Alle Prozesse sind gleichwertig
  - einzig Hinweis auf eine Prozesshierarchie ist ein spezielles Token (sog. *Handle*), das ein Elternprozess bei der Erzeugung eines Prozesses erhält, um seinen Kindprozess zu steuern.
    - Token kann allerdings an einen anderen Prozess weitergegeben werden.

Prozesse unter Windows

- Welche Werkzeuge bietet Windows zur Steuerung von Prozessen? Benennen Sie die wichtigsten Optionen.
  - Prozess starten
  - Prozess anzeigen
  - Prozess beenden

Prozesszustände

- running - rechnend, wird auf der CPU **ausgeführt**
- ready – rechenbereit, **kurzzeitig gestoppt**, um einen anderen Prozess rechnen zu lassen
- blocked – blockiert, **nicht lauffähig** bis ein best. externes Ereignis eintritt
  - Bspw. „Pause"; keine Tastatureingabe



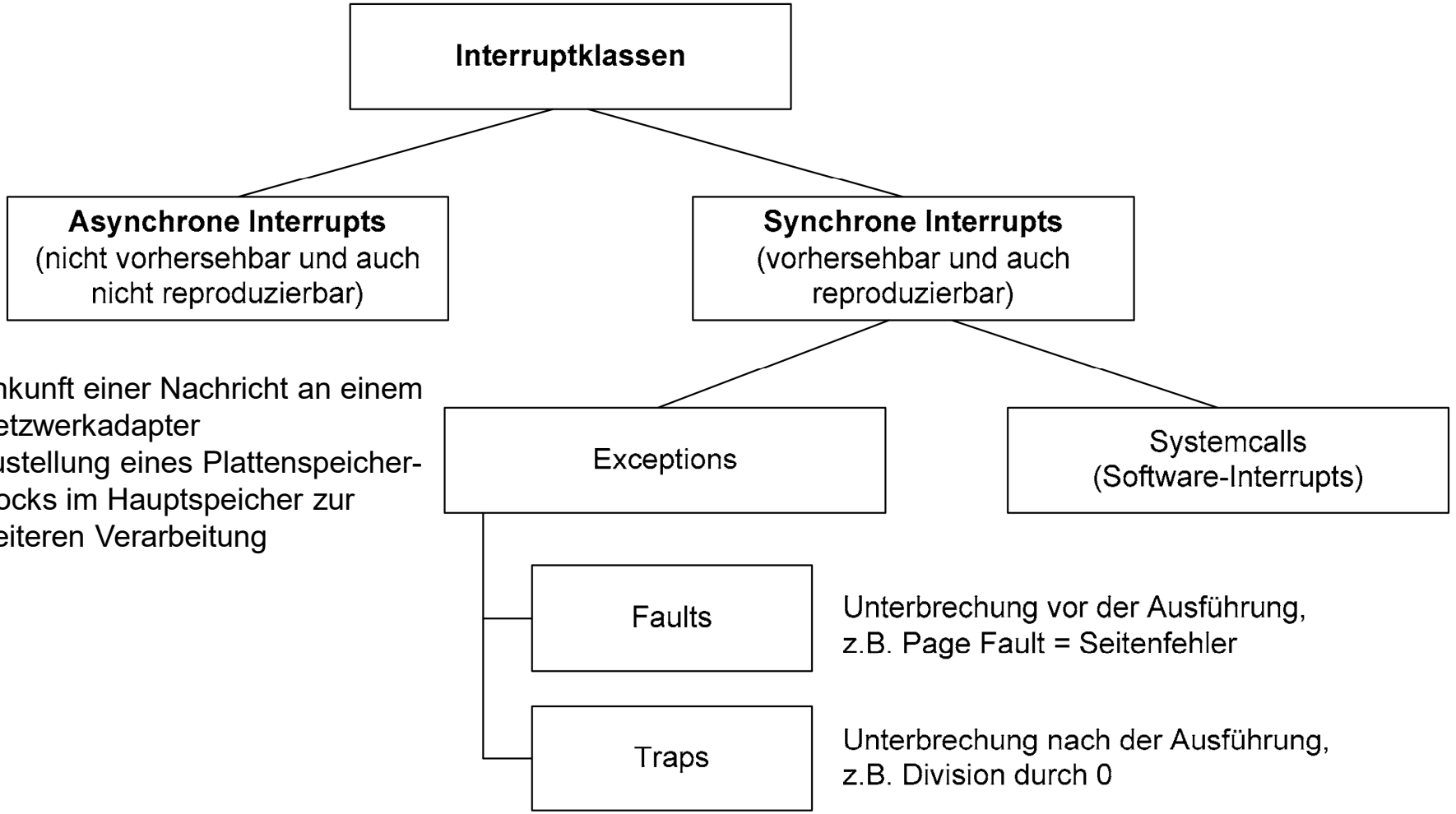
1. Prozess blockiert, weil er auf Eingabe wartet
2. Scheduler wählt einen anderen Prozess aus
3. Scheduler wählt diesen Prozess aus
4. Eingabe vorhanden

Interrupt

Im Gegensatz zu Polling sind Interrupts (Unterbrechungen) sog. Betriebssystembedingungen oder auch asynchrone Ereignisse, die den Prozessor veranlassen, einen vordefinierten Code auszuführen, der außerhalb des normalen Programmflusses liegt. (Russinovich)

- Durch Hardware oder Software verursacht

Interrupt



Bsp:

- Ankunft einer Nachricht an einem Netzwerkadapter
- Zustellung eines Plattenspeicher-blocks im Hauptspeicher zur weiteren Verarbeitung

Interrupt-Bearbeitung

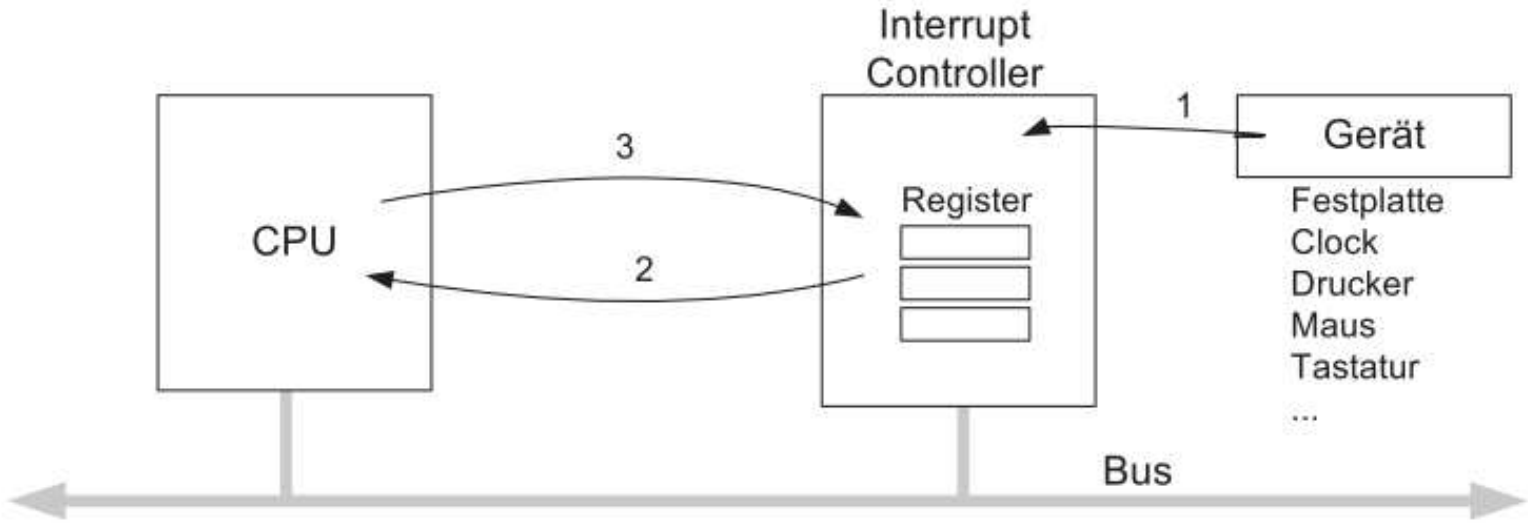
- Steuerung an eine definierte Position im Kernel übergeben

  -> Wechsel von Benutzermodus in Kernelmodus

- Maskierung
  - Ein-/Ausschalten von Interrupts für E/A-Geräte
    - IMR (Interrupt Mask Register)
    - Bit auf 1 -> ausgeschaltet
  - !Aber auch NMI (Non Maskable Interrupt) -> schwerwiegende Ausnahmesituation
- Interrupt-Service-Routine (ISR)
  - Programmstück, für die Interrupt-Bearbeitung
  - Übernimmt Kontrolle nach Unterbrechung der Programmausführung

Interrupt-Bearbeitung

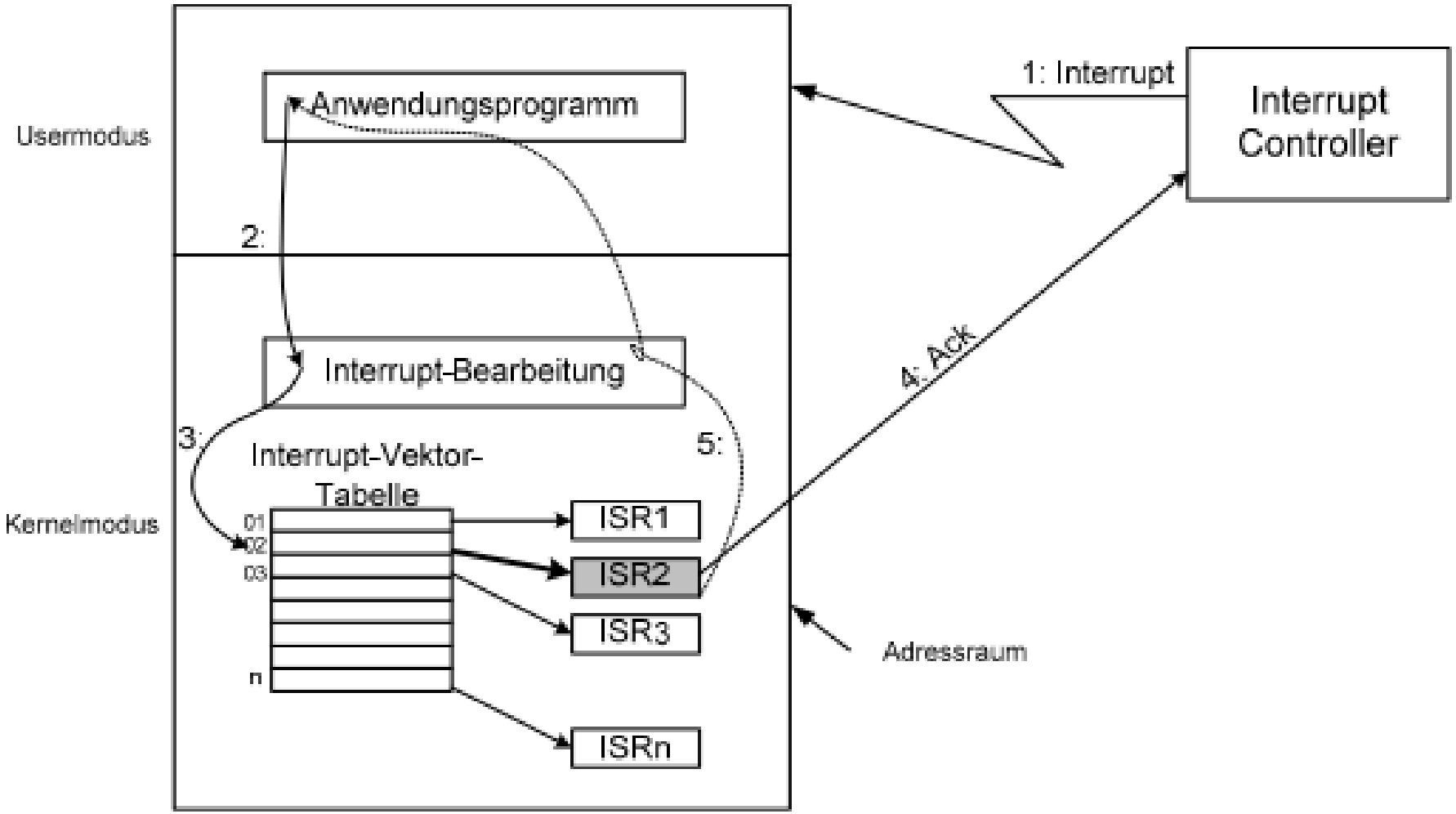
- Interrupt-Request-Bearbeitung (IRQ-Bearbeitung)
  - Hardwarebedingte Interrupts werden i.d.R. nicht direkt an CPU gesendet
  - Zunächst an Interrupt-Controller durch Interrupt-Request
  - Controller erzeugt dann Unterbrechung der CPU mit ISR (*interrupt service routine)*



1: Das Gerät ist mit seiner Arbeit fertig (z.B. Holen eines Datenblocks aus einer Festplatte)
2: Der Interrupt Controller erzeugt einen Interrupt
3: Die CPU bestätigt den Interrupt

Interrupt-Bearbeitung

- Interrupt-Vektor-Tabelle
  - ISRs werden in heutigen Systemen über einen Interrupt-Vektor adressiert
  - Enthält die Speicheradresse der Interrupt-Service-Routine
- Interrupt-Level
  - Priorität im Gesamtverarbeitungsprozess
  - Reihenfolge bei gleichzeitig auftretenden Interrupts

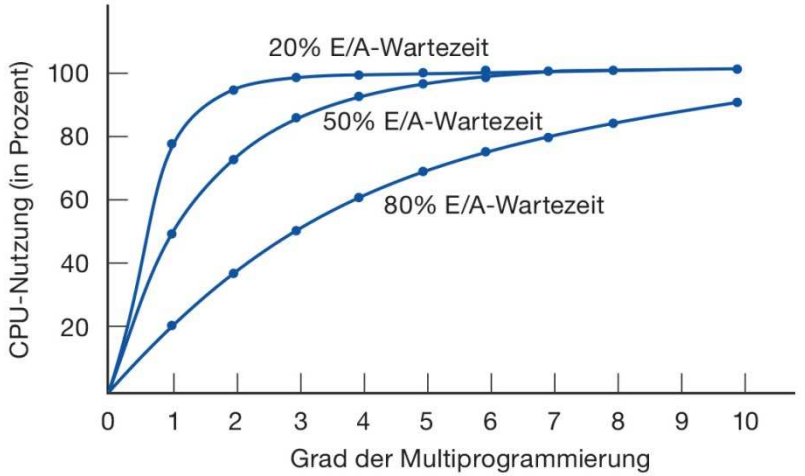

Usermodus
Kernelmodus
Anwendungsprogramm
2:
Interrupt-Bearbeitung
3:
Interrupt-Vektor-
Tabelle
01
02
03
n
ISR1
ISR2
ISR3
ISRn
5:
1: Interrupt
Interrupt
Controller
4: Ack
Adressraum

Systemcalls / Systemaufrufe

- Dienstaufruf an das Betriebssystem, bei dessen Ausführung in den Kernelmodus gewechselt wird.
- Kontrollfluss wird dabei meist von einem Anwendungs-programm an den Kernel übergeben.
- Beispiel Windows
  - CreateProcess zur Prozesserzeugung
  - ExitProcess zum Beenden eines Prozesses
  - CreateFile zum Erzeugen einer Datei
  - ReadFile zum Lesen einer Datei

Grad der Multiprogrammierung

- Annahme: n Prozesse, p Wartezeit für E/A [0,1]
- CPU-Ausnutzung = $1-p^n$
- Diagramm gibt an,
  - wie viele Prozesses im Speicher sein müssen
  - um bei gegebener E/A-Wartezeit
  - die gewünschte CPU-Auslastung zu erreichen



CPU-Ausnutzung als eine Funktion der Anzahl der Prozesse im Speicher.

Rechenbeispiel

- PC mit 512 MB Hauptspeicher
  - o 128 MB für das Betriebssystem
  - o 128 MB für Benutzerprozesse
  - o typische Büroanwendungen, d. h. 80% E/A-Wartezeit
  - ? Programme im Speicher:
  - ? Auslastung:
- Aufrüstung um 512 MB
  - ? Programme im Speicher:
  - ? Auslastung:
- Nochmals Aufrüstung um weitere 512 MB
  - ? Programme im Speicher:
  - ? Auslastung:

Trap-Mechanismus

- Ein Trap ist ein Software-Interrupt, der für einen Übergang vom Benutzermodus in den Kernelmodus sorgt, um den aufgerufenen Systemdienst auszuführen ohne die Adressen von Systemroutinen kennen zu müssen.

Aufgaben

1. Was ist der Unterschied zwischen Polling und interruptgesteuerter Verarbeitung?
2. Was ist der Unterschied zwischen den Exception-Typen Fault und Trap? Nennen Sie jeweils ein Beispiel!
3. Was bedeutet „Maskierung“ von Unterbrechungsanforderungen?
4. Wie erkennt die CPU, dass eine Unterbrechungsanforderung ansteht?
5. Was versteht man unter einer Interrupt-Vektor-Tabelle?
6. Was ist eine Interrupt-Service-Routine und wann wird sie aufgerufen?
7. Nennen Sie den Unterschied zwischen einem synchronen und asynchronen Interrupt!

Aufgaben

1) Ein Computersystem hat genügend Platz, um fünf Programme in seinem Arbeitsspeicher zu halten. Diese Programme befinden sich die Hälfte der Zeit im Leerlauf, weil sie auf Ein-/Ausgabe warten. Wie groß ist der Anteil an CPU-Zeit, der verschwendet wird?

2) Ein Rechner verfügt über 4 GB RAM, wovon das Betriebssystem 512 MB belegt. Die Prozesse benötigen (der Einfachheit halber) alle jeweils 256 MB und haben dieselben Eigenschaften. Falls das Ziel eine 99%ige CPU-Ausnutzung ist, welches ist die maximale Ein-/Ausgabewartezeit, die toleriert werden kann?

3) Mehrere Aufträge können schneller ausgeführt werden, wenn sie parallel statt sequenziell ablaufen. Angenommen, zwei Jobs starten gleichzeitig, wobei jeder 20 Minuten CPU-Zeit benötigt. Wie lange wird es dauern, bis der letzte Auftrag beendet ist, wenn sie sequenziell ablaufen? Und wie viel Zeit vergeht, wenn sie parallel laufen? Setzen Sie eine Ein-/Ausgabewartezeit von 50 % voraus.

Aufgabe 1

Ein Computersystem hat genügend Platz, um fünf Programme in seinem Arbeitsspeicher zu halten. Diese Programme befinden sich die Hälfte der Zeit im Leerlauf, weil sie auf Ein-/Ausgabe warten. Wie groß ist der Anteil an CPU-Zeit, der verschwendet wird?

Aufgabe 2

Ein Rechner verfügt über 4 GB RAM, wovon das Betriebssystem 512 MB belegt. Die Prozesse benötigen (der Einfachheit halber) alle jeweils 256 MB und haben dieselben Eigenschaften. Falls das Ziel eine 99%ige CPU-Ausnutzung ist, welches ist die maximale Ein-/Ausgabewartezeit, die toleriert werden kann?

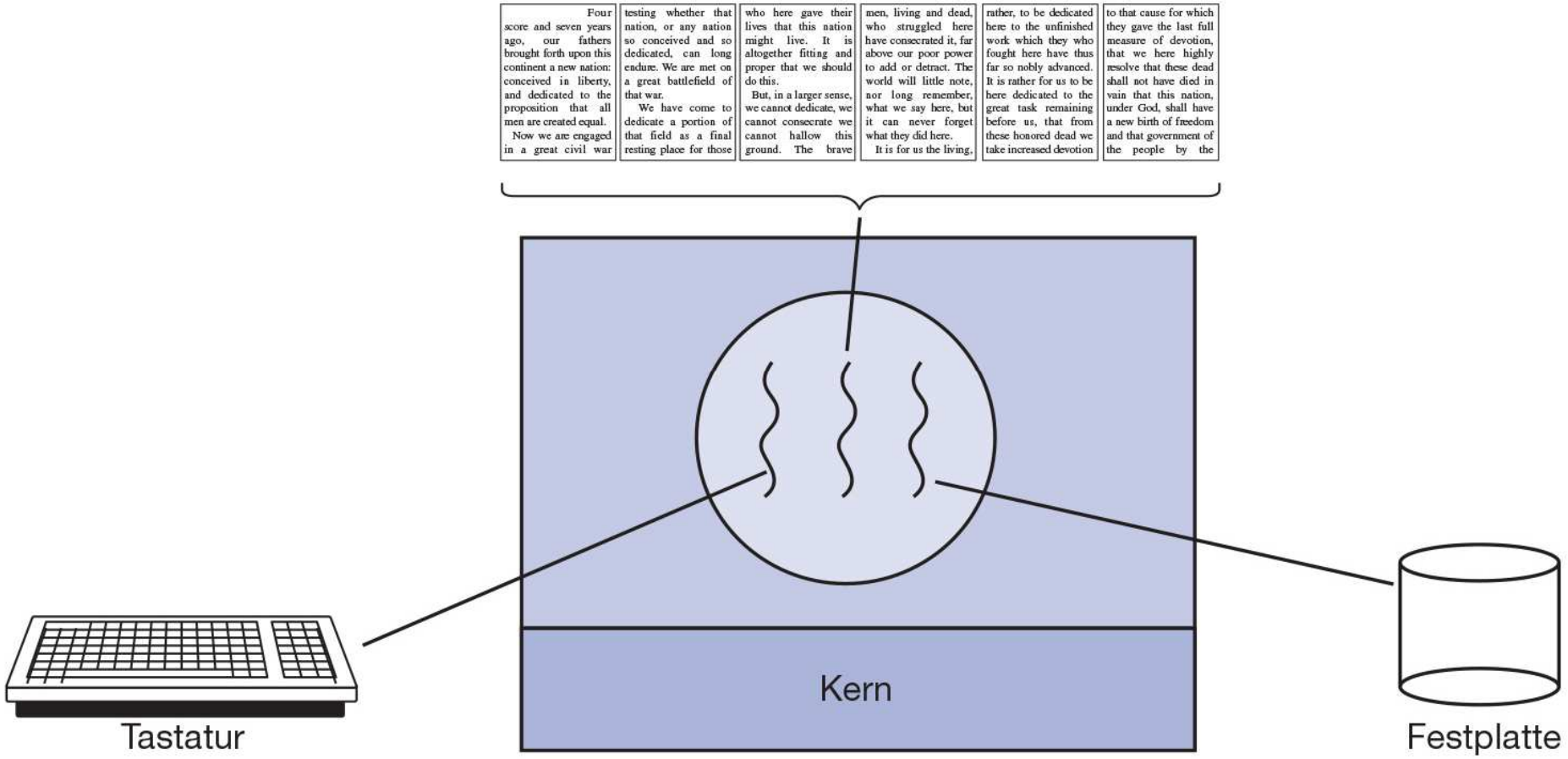
Aufgabe 3

Mehrere Aufträge können schneller ausgeführt werden, wenn sie parallel statt sequenziell ablaufen. Angenommen, zwei Jobs starten gleichzeitig, wobei jeder 20 Minuten CPU-Zeit benötigt. Wie lange wird es dauern, bis der letzte Auftrag beendet ist, wenn sie sequenziell ablaufen? Und wie viel Zeit vergeht, wenn sie parallel laufen? Setzen Sie eine Ein-/Ausgabewartezeit von 50 % voraus.

Threads

- Prozess
  - Eigener Adressraum
  - Einen einzigen Ausführungsfaden (**thread** of control)
- Threads dienen zur einfacheren Programmierung
  - Mehrere Threads des selben Prozesses greifen auf den selben Adressraum mit all seinen Daten zu.
  - Leichtgewichtiger als Prozesse („Miniprozesse"); schneller zu erzeugen und zu beenden (10 bis 100-mal schneller)
  - Performanter bei umfangreichen Ein-/Ausgabeaktivitäten
  - Nützlicher bei mehreren CPUs mit echter Parallelität

Threads
Bsp. Textverarbeitung



Threads
Bsp. Webserver

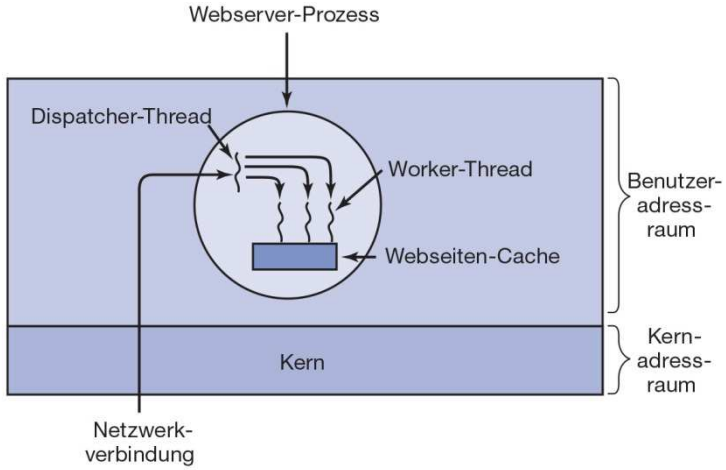


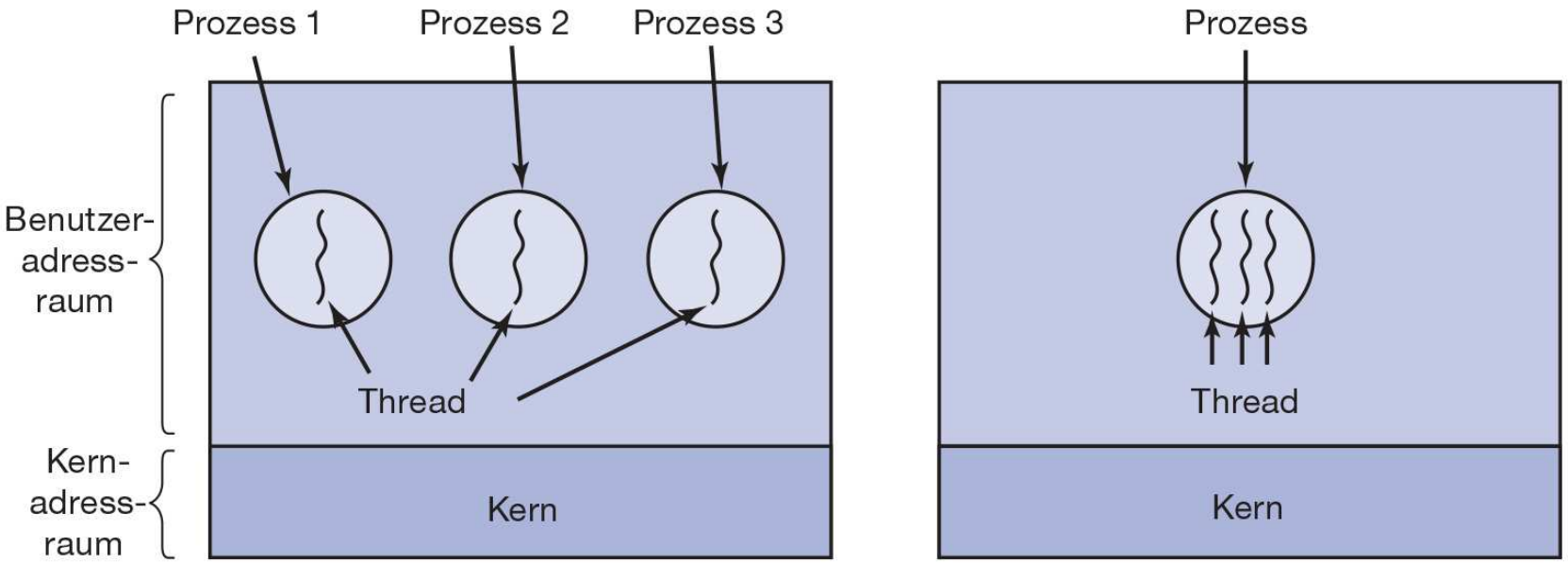
Abbildung 2.8: Ein Webserver mit mehreren Threads.

- Dispatcher-Thread liest Anfragen aus dem Netz ein
- wählt einen unbeschäftigten Worker-Thread um Anfrage zu bearbeiten
- Worker-Thread bearbeitet Anfrage
  - aus dem Cache, falls vorhanden -> kurze Antwortzeit
  - von der Festplatte, ansonsten -> große Antwortzeit
  - Zugriff auf Festplatte ist E/A, Thread blockiert
  - anderer Thread kann starten (z. B. Dispatcher-Thread oder weiterer Worker-Thread)

Klassische Threadmodell

- Threads erweitern Prozessmodell um:
  - Mehrere Ausführungsfäden
  - In hohem Grad unabhängig voneinander
  - In der selben Prozessumgebung.
- Viele Threads parallel in einem Prozess ist analog zu mehreren Prozessen auf einem Rechner
  - Prozesse teilen sich physischen Speicher, Festplatten, Drucker und andere Betriebsmittel.
  - Threads teilen sich einen Adressraum und andere Ressourcen
- Mehrere Threads in einem Prozess -> Multithreading
  - Einige CPUs haben direkte Hardwareunterstützung für Multithreading! (Thread-Wechsel im Nanosekundenbereich)

Threads


Prozess 1
Prozess 2
Prozess 3
Benutzer-
adress-
raum
Thread
Kern-
adress-
raum
Kern
Prozess
Thread
Kern

Threads

- Alles Threads nutzen den gleichen Adressraum
- Gleiche globale Variablen
- Jeder Thread kann den Stack des anderen lesen, schreiben oder löschen
- Kein Schutz zwischen den Stacks
  - Auch nicht notwendig, da kooperierend und nicht konkurrierend
- Teilen sich auch gleiche geöffnete Dateien, Alarme, Signale

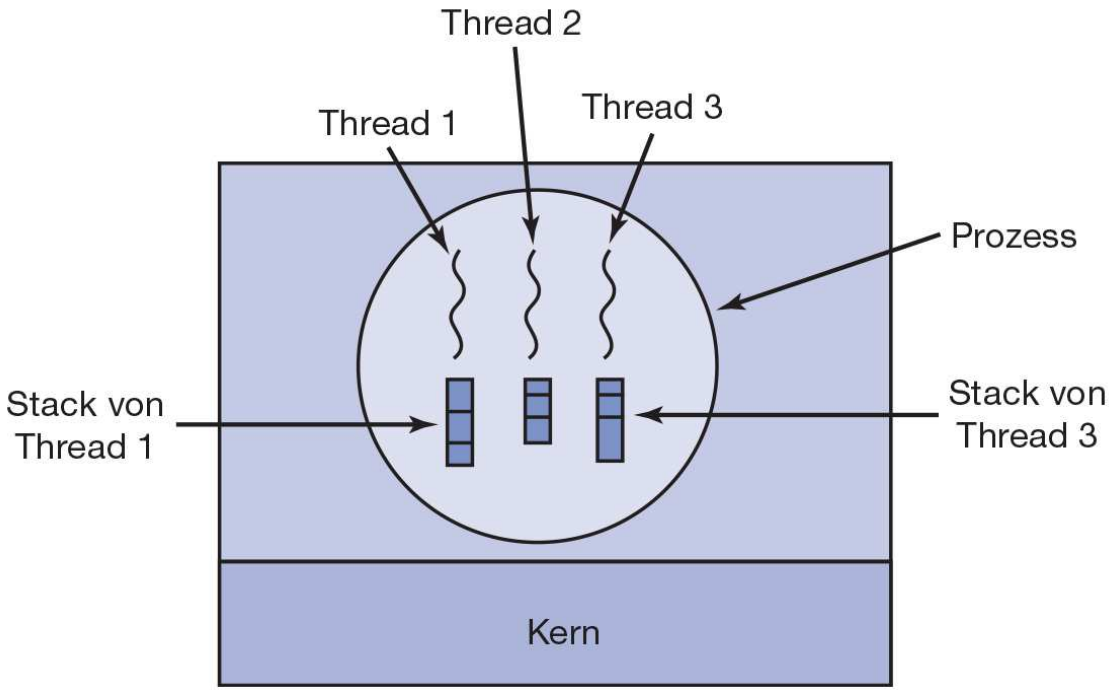
Threads

Elemente pro Prozess	Elemente pro Thread
Adressraum	Befehlszähler
Globale Variable	Register
Geöffnete Dateien	Stack
Kindprozesse	Zustand
Ausstehende Signale	
Signale und Signalroutinen	
Verwaltungsinformationen	

- **Elemente pro Prozess**: die Elemente, die von Threads **geteilt** werden.
- **Elemente pro Thread**: die Elemente, die **individuell** für einen Thread sind.

Threads

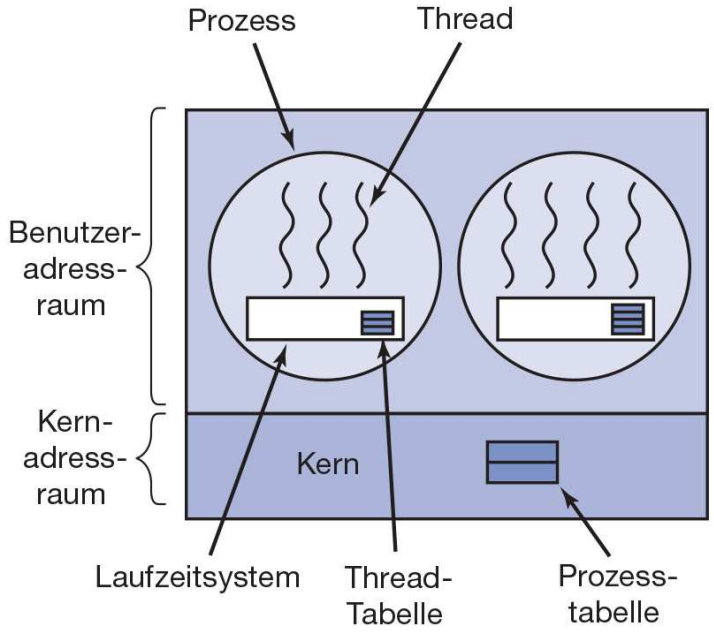
- Zustände wie Prozessmodell
  - rechnend – blockiert – rechenbereit
- jeder Thread hat eigenen Stack



Threads im Benutzeradressraum

- Verwaltung der Threads durch Prozess
  - jeder Prozess hat eigene Threadtabelle

- Vorteil
  - sehr schnelle Threadwechsel ohne Kontextwechsel
  - individuelle Scheduling-Algorithmen können eingesetzt werden
- Nachteil
  - alle Threads werden bei E/A-Warten blockiert
  - Threads müssen Prozessor freiwillig abgeben

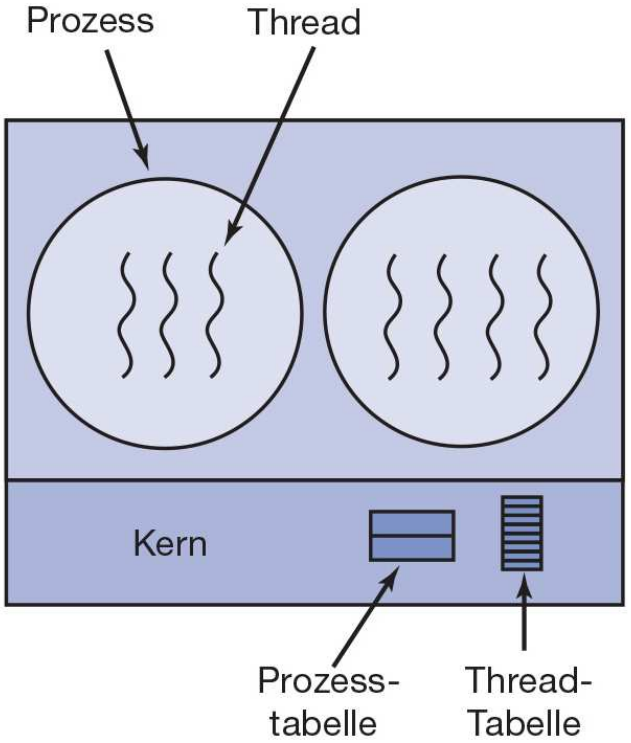


Threads im Kernel

- Verwaltung von Threads durch Kernel

- Vorteil
  - kontrolliert vom Betriebssystem
  - eine Thread-Tabelle
  - E/A ist thread-unabhängig

- Nachteil
  - Threadwechsel aufwendig
  - Threaderzeugung und –beendigung aufwendig da nur durch Betriebs-systemfunktionen realisierbar



Gründe für Verwendung von Threads

- **Responsivität der Benutzeroberfläche**: die grafische Oberfläche soll bedienbar bleiben, während im Hintergrund eine rechenintensive Aktivität ausgeführt wird
- **Asynchrones Warten**: Überbrückung von Wartezeiten, z.B. bei der asynchronen Ein- und Ausgabe, indem andere Aktivitäten fortgeführt werden. Webbrowser: Daten empfangen und Webseite rendern
- **Trennung von Teilaufgaben**: Voneinander unabhängige Aktivitäten sollen implementiert werden (Sound, Animation, Logik bei einem Computerspiel)
- **Bei Mehrkernsystemen**: Threads bilden die kleinste Einheit paralleler Aktivität. Aufgaben können zur Geschwindigkeits-steigerung parallelisiert werden

Threads
Hybride Implementierung

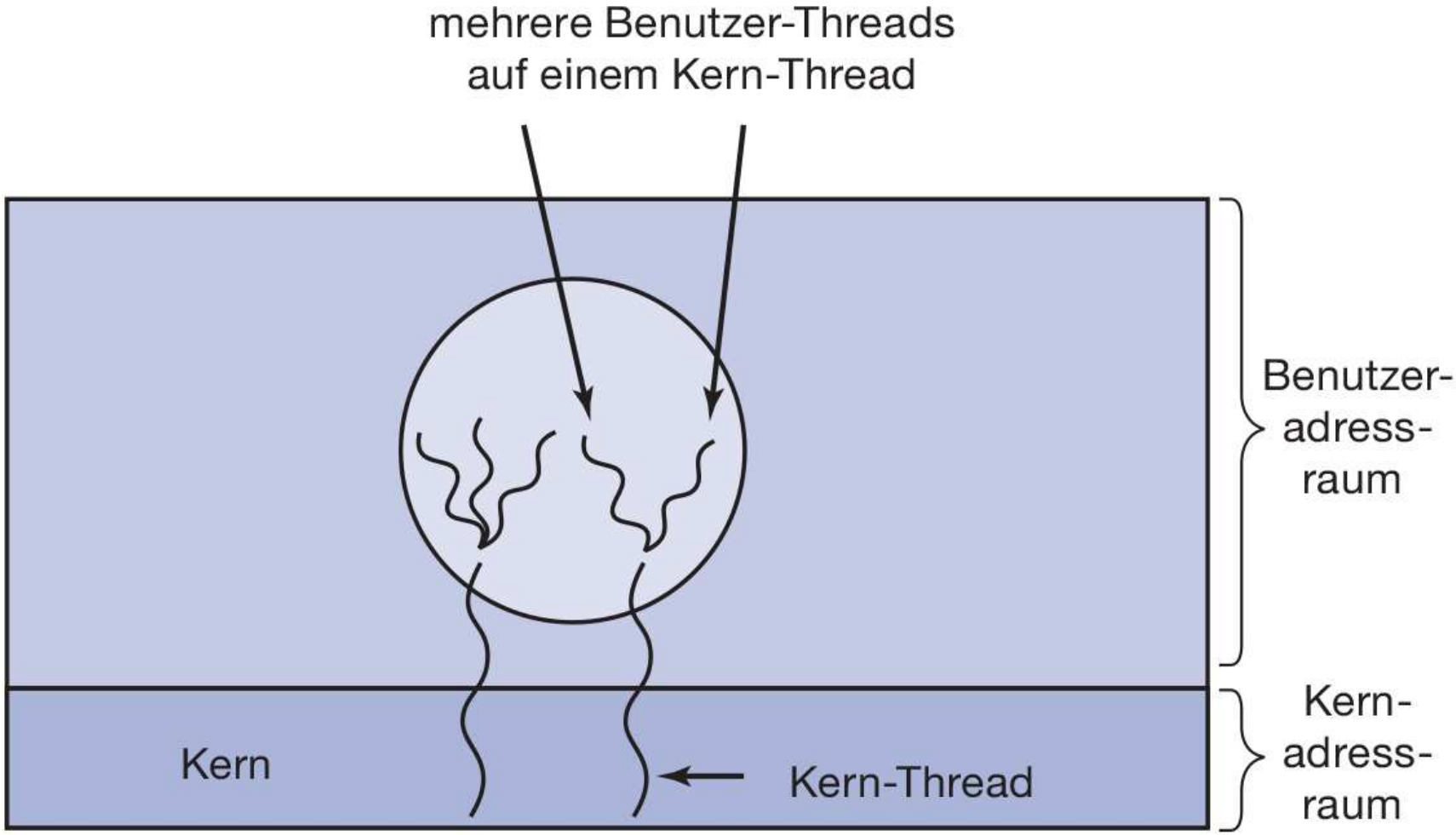


Abbildung 2.17: Bündeln von Benutzer-Threads auf Kern-Threads.

Aufgaben

1. Wie verhalten sich Threads zu Prozessen im Hinblick auf die Nutzung des Prozessadressraums?
2. Was bedeutet eine 1:n-Beziehung zwischen den Betriebsmitteln Prozess und Thread?
3. Was ist in der Prozessverwaltung ein PCB, wozu dient er und welche Inhalte hat er? Nennen Sie dabei drei wichtige Informationen, die im PCB verwaltet werden!
4. Threads werden heute von den meisten Betriebssystemen unterstützt. Was versteht man unter einem Thread?
5. Beschreiben Sie einen einfachen Zustandsautomaten eines Prozesses!