Betriebssysteme

Interruptverarbeitung

Literatur Verzeichnis

- Mandl, Peter; Grundkurs Betriebssysteme; 5.Aufl. 2020; Springer Verlag
- M.Russinovich, D.A.Solomon,A.Iomescu; Windows Internal Part 1; 6 Auflage, Microsoft Press 2012

Gliederung

- Begriffe der Interruptverarbeitung
- Bearbeitung und Abläufe von Interrupts
- Beispiel eines Hardwarebausteins zur Interruptverarbeitung
- Die verschiedenen Interrupt-Klassen
- Ablauf eines asynchronen Interrupts
- Ablauf eines (synchronen) Systemcalls

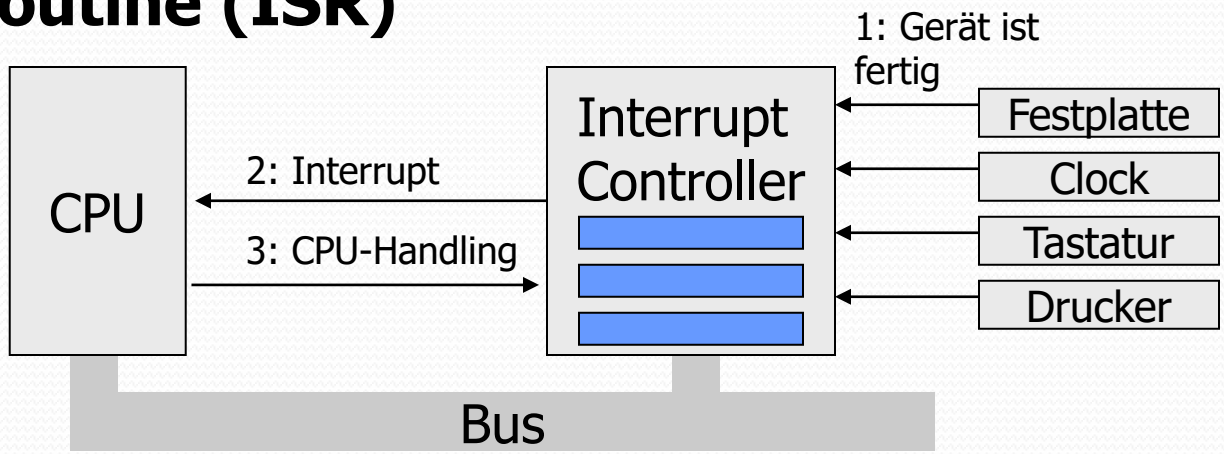
Aufgabe

- Erläutern Sie folgende Begriffe und geben Sie dazu aussagekräftige Beispiele an:
  - Polling
  - Synchrone Interrupts
  - Asynchrone Interrupts
  - Maskierden von Interrupts

Interrupt

Abläufe I

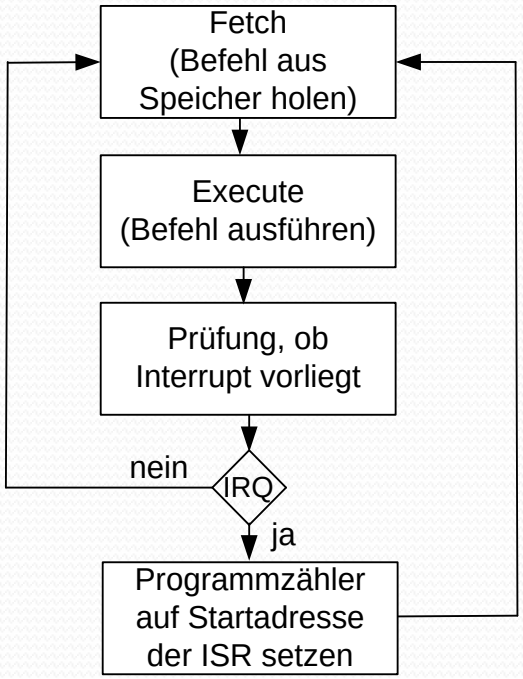
- Interrupts führen dazu, dass Code außerhalb des normalen Programmflusses ausgeführt wird
- Steuerung wird an eine definierte Position im Kernel übergeben → **Interrupt-Service-Routine (ISR)**



Interrupt

Abläufe II

- Prüfung, ob Interrupt anliegt, ist Teil des Befehlszyklus
- Prüfung am Ende eines Maschinenbefehls

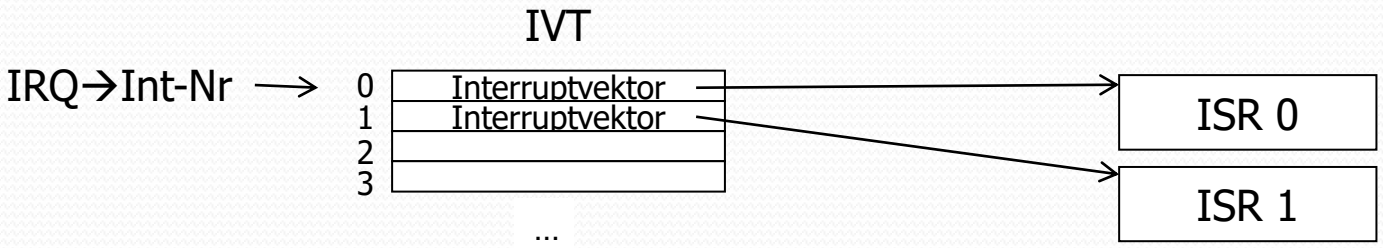


- Bei Multiprozessoren bzw. Mehrkernprozessoren:
  - Dispatching eines Prozessors/Kerns notwendig, um anstehenden Interrupt zu bearbeiten

Interrupt

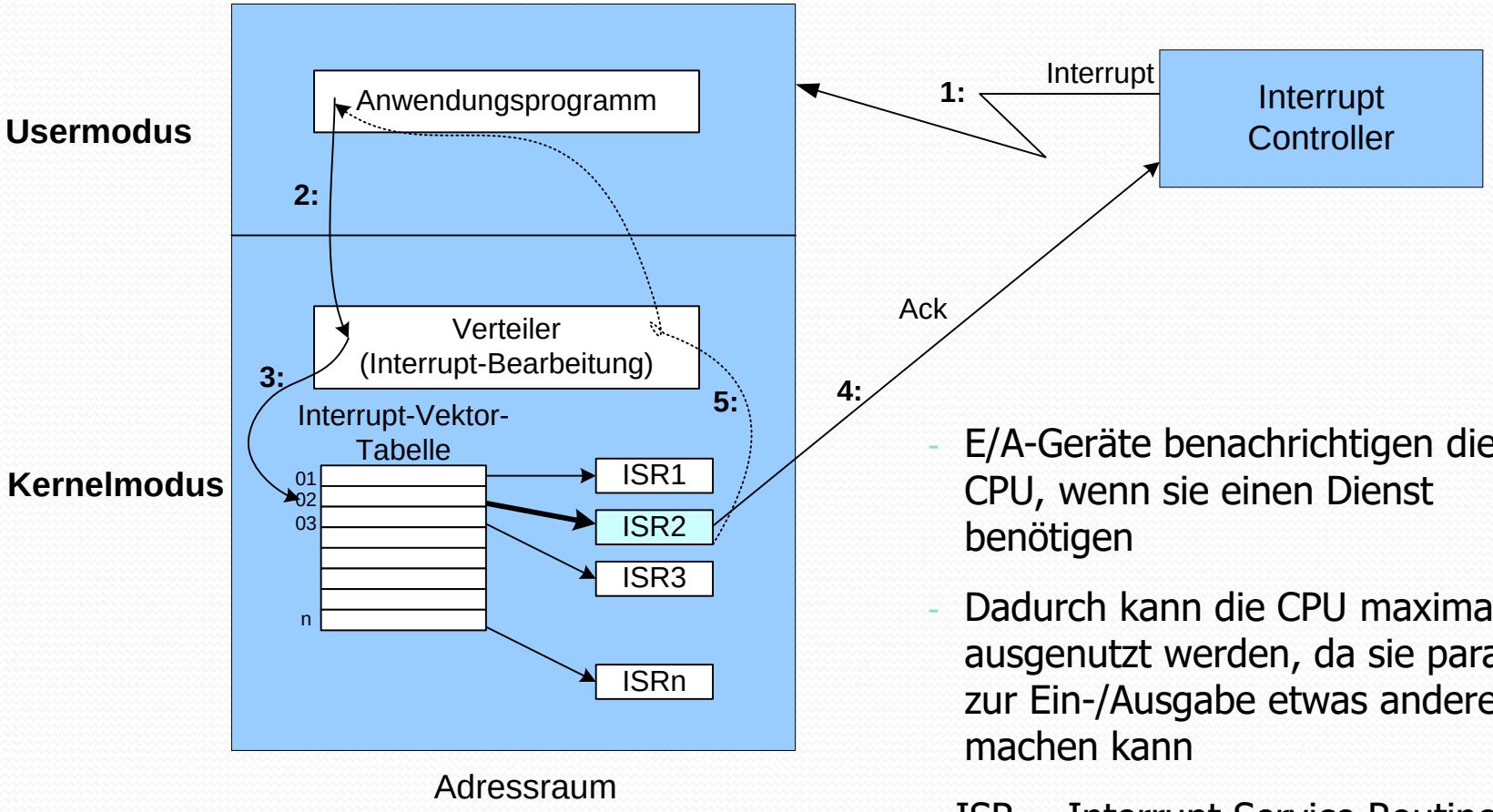
Vektor-Tabelle und Adressierung

- Interrupt Request (IRQ) wird vom Gerät gesendet und identifiziert das Gerät
- Abbildung IRQ → Int-Nr durch Hardware in einem Interrupt Controller
  - Int-Nr ist der Index für die Interrupt-Vektor-Tabelle (IVT)
  - Die IVT wird über die CPU adressiert
- IVT-Aufbau durch Prozessor vorgegeben:
  - Bei Intel 256 IVT-Einträge für Exceptions, Systemcalls (Traps) und Geräteinterrupts



Interrupt prinzipiell

Service-Routine (ISR)

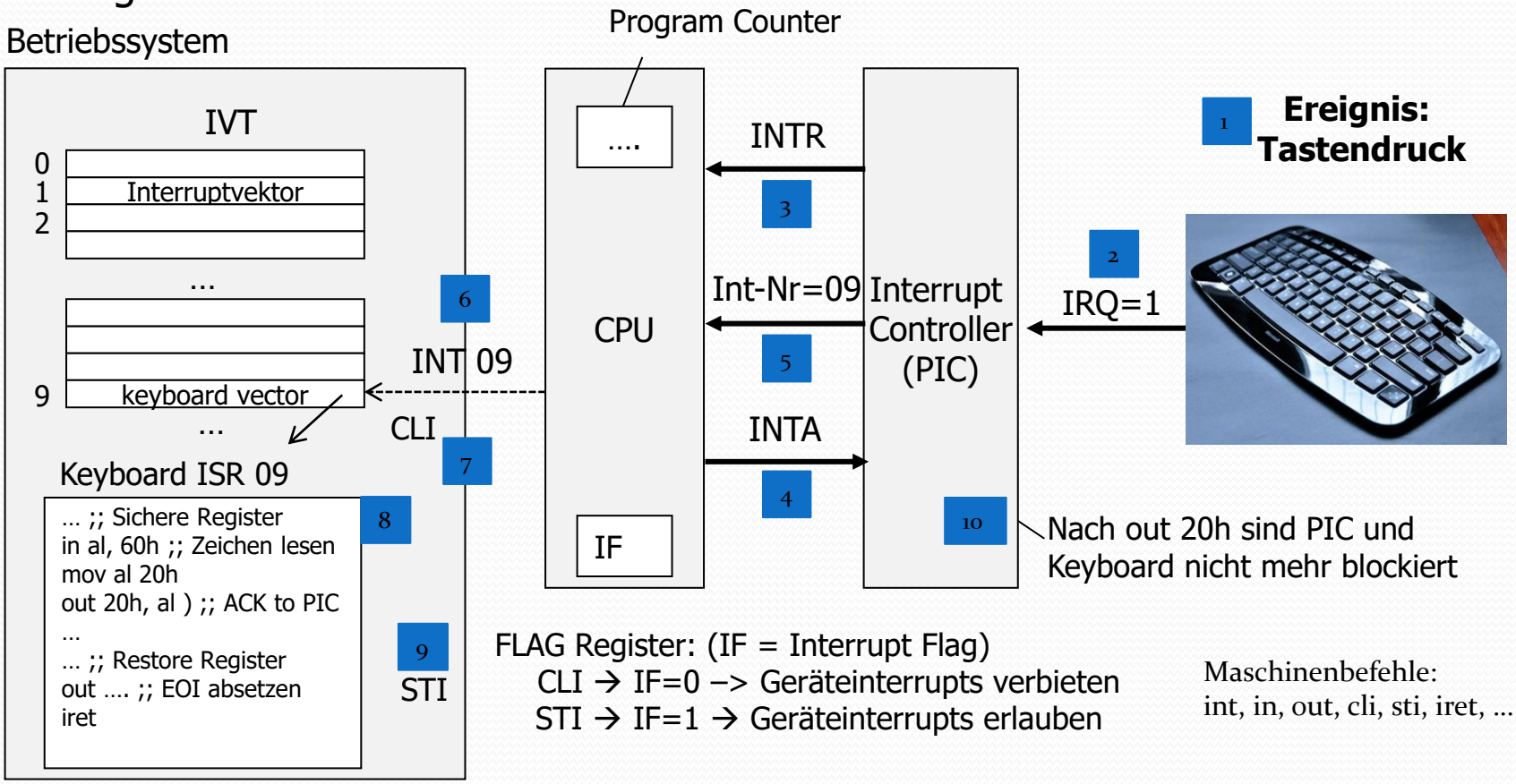


- E/A-Geräte benachrichtigen die CPU, wenn sie einen Dienst benötigen
- Dadurch kann die CPU maximal ausgenutzt werden, da sie parallel zur Ein-/Ausgabe etwas anderes machen kann

ISR = Interrupt Service Routine

Beispiel: Keyboard-Interrupt

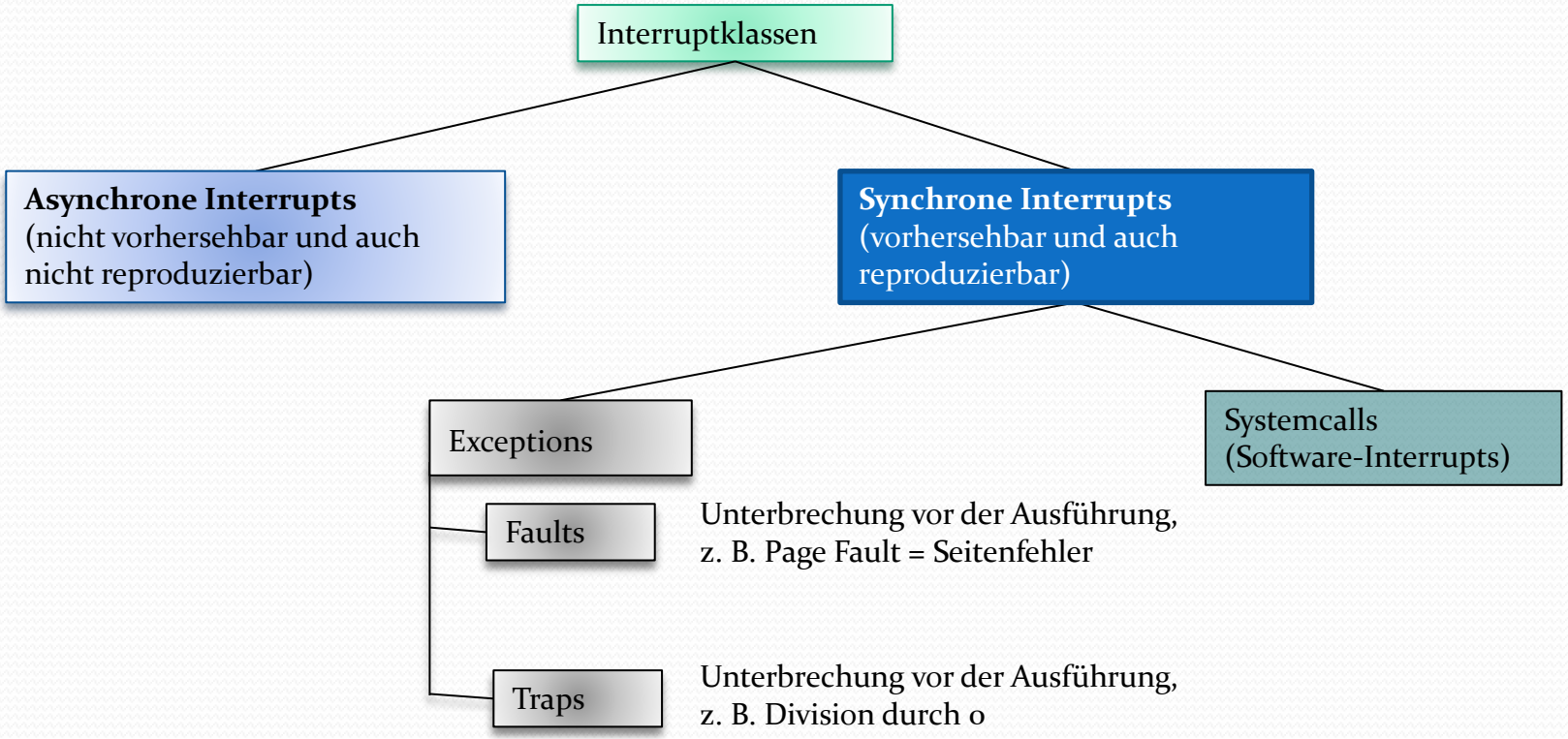
- IRQ = 1 → Keyboard (Tastatur)
- PIC und Keyboard sind bis zum ACK-Signal blockiert → es muss schnell gehen!



FLAG Register: (IF = Interrupt Flag)
CLI → IF=0 –> Geräteinterrupts verbieten
STI → IF=1 → Geräteinterrupts erlauben

Maschinenbefehle:
int, in, out, cli, sti, iret, ...

Interrupt-Klassen



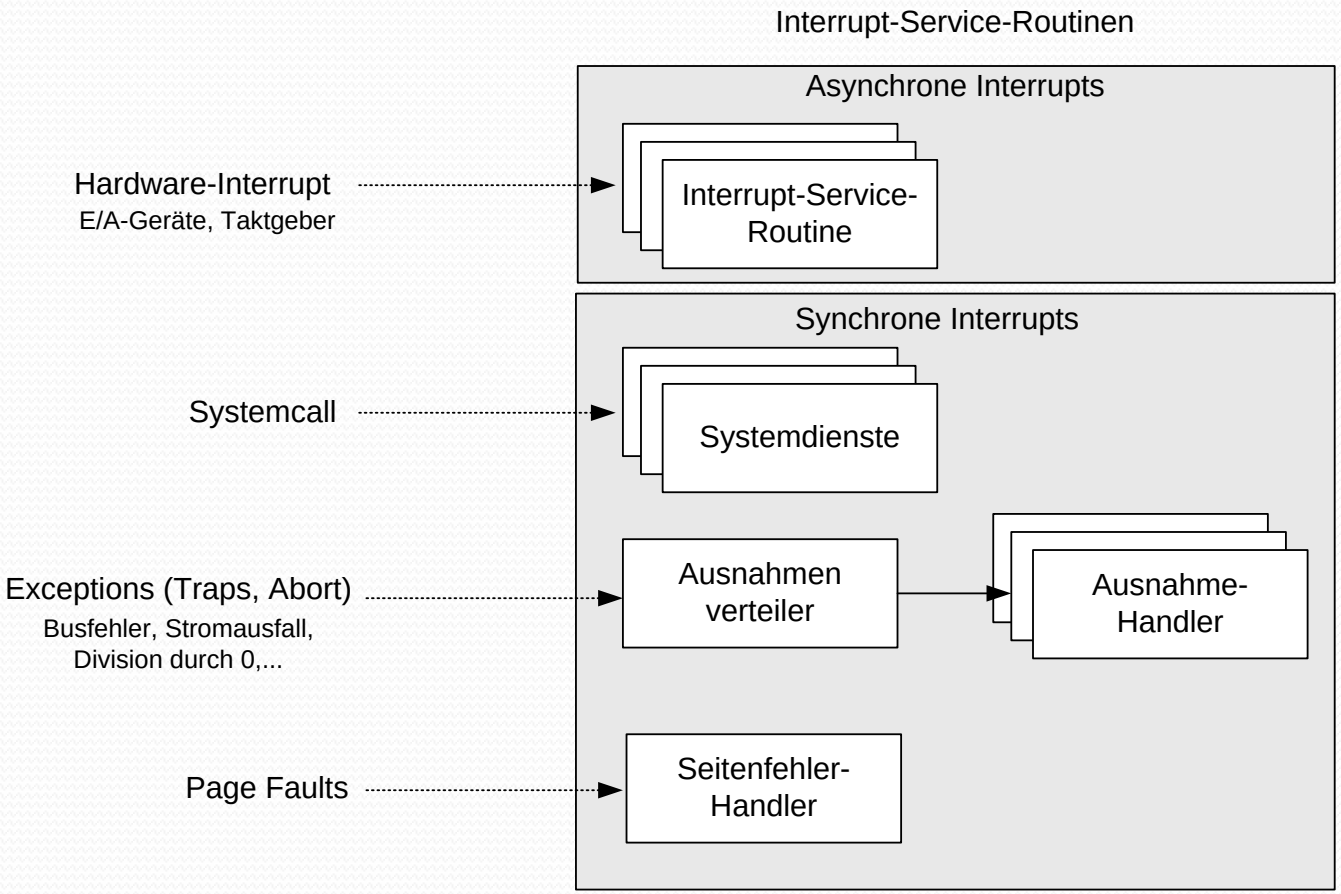
Einschub:

Memory-Mapped I/O versus Port-Mapped I/O

- Zwei Verfahren zur Kommunikation der CPU mit externen Geräten bzw. Gerätecontrollern (z.B. PIC/APIC)
- Memory-Mapped I/O:
  - Die Register von Gerätecontrollern werden auf Hauptspeicheradressen abgebildet
  - Jeder Controller kennt seine Adressen und bearbeitet den Zugriff, falls eine seiner Adressen auf dem Adressbus liegt
- Port-Mapped I/O:
  - Die Register von Controllern werden über eigene Portadressen in einem eigenen I/P-Adressraum angesprochen
  - Eigene Signalleitung wird verwendet, um jeweiligen Adressraum zu adressieren (Arbeitsspeicher oder I/O-Speicher)

Beispiel Windows

Interrupt-Bearbeitung (1)



Beispiel Windows

Interrupt-Bearbeitung (2)

- Windows hat eine eigene Interrupt-Verwaltung
- Über sog. **Interrupt Request Levels** (IRQL) ordnet der Kernel den Interrupts eigene Prioritäten zu
- Nur Interrupts mit höherem IRQL können Interruptbearbeitung auf niedrigerem IRQL unterbrechen
- Über eine Interrupt Dispatch Tabelle (IDT) wird festgehalten, welche ISR für welchen Interrupt zuständig ist

Beispiel Windows

Interrupt-Bearbeitung (3)

- Interrupt Request Levels (IRQLs) in der IA32-Architektur

IRQL	Bezeichnung	Beschreibung
31	High-Level	Maschinen-Check und katastrophale Fehler
30	Power-Level	Strom/Spannungsproblem
29	IPI-Level	Interprocessor Interrupt
28	Clock-Level	Clock-Interrupt
27	Sync-Level	Prozessorübergreifende Synchronisation
3-26	Device-Levels	Abbildung auf IRQs der Geräte je nach verbautem Interrupt-Controller
2	Dispatch/DPC-Level	Dispatching und Ausführung von Deferred Procedure Calls
1	APC-Level	Ausführung von Asynchronous Procedure Calls nach Ein-/Ausgabe-Requests
0	Passive-Level	Normale Threadausführung

Beispiel Windows

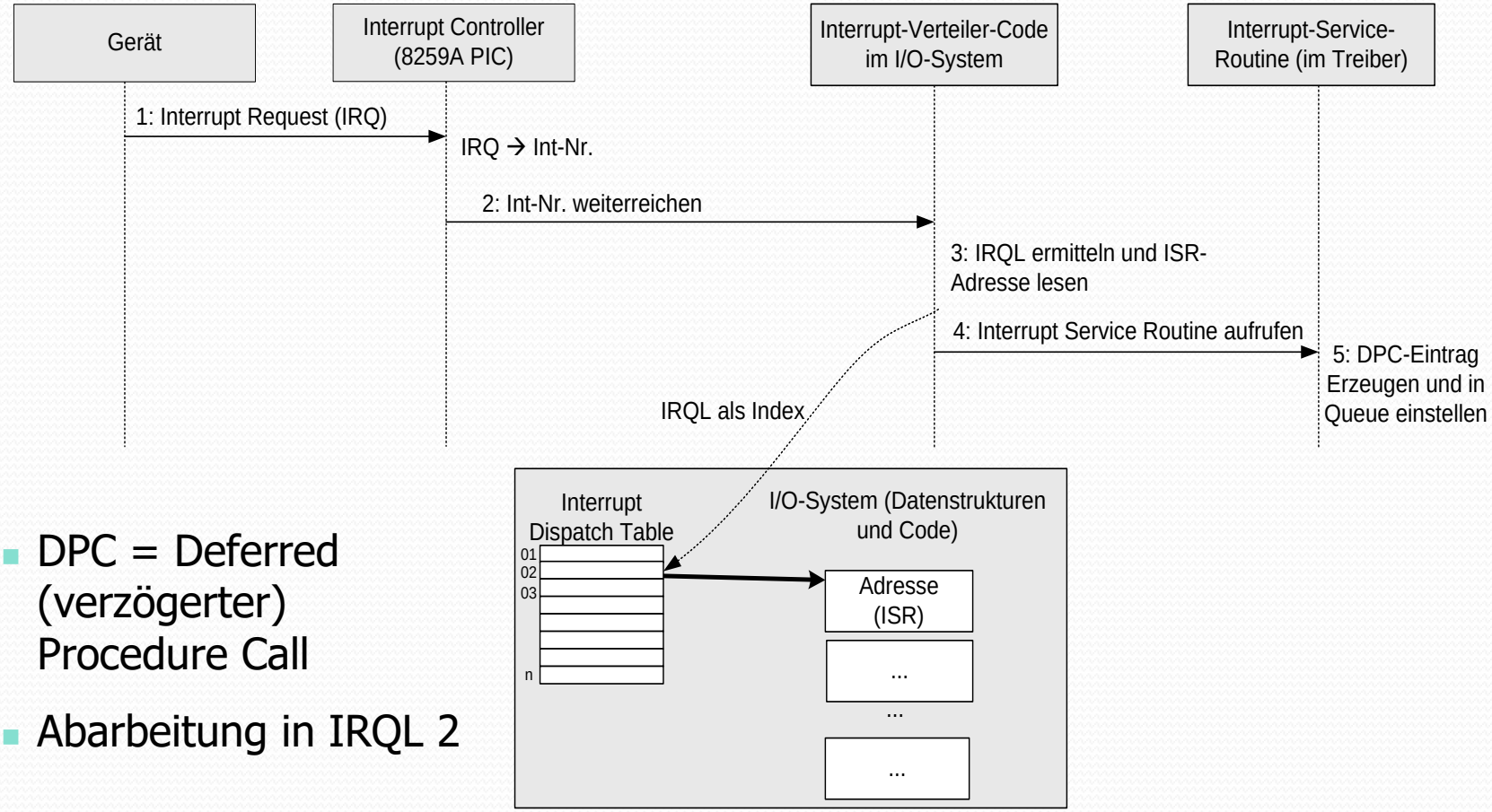
Interrupt-Bearbeitung (4)

- Interrupt Request Levels (IRQLs) in der x64- und der IA64-Architektur

IRQL	Bezeichnung	Beschreibung
15	High-Level	Maschinen-Check und katastrophale Fehler
14	Power-Level	Strom/Spannungsproblem und Interprozessor-Interrupt
13	Clock-Level	Clock-Interrupt
12	Synch-Level	Prozessorübergreifende Synchronisation
3-11	Device-Levels	Abbildung auf IRQs der Geräte je nach verbautem Interrupt-Controller
2	Dispatch/DPC-Level	Dispatching und Ausführung von Deferred Procedure Calls
1	APC-Level	Ausführung von Asynchronous Procedure Calls
0	Passive-Level	Normale Threadausführung

Beispiel Windows

Interrupt-Bearbeitung (5)



- DPC = Deferred (verzögerter) Procedure Call
- Abarbeitung in IRQL 2

Beispiel Linux

Interrupt (1)

- Linux nutzt auch eine Tabelle mit Referenzen auf die Interrupt-Handler (ISR)
- Jeder Interrupt-Request wird auf eine Interrupt-Nummer (= Index in der Tabelle) abgebildet
- Meist wird in der ISR nur ein **Tasklet** erzeugt
- Tasklets dienen der schnellen Behandlung von Interrupts (ähnlich dem Windows-DPC-Mechanismus)

Beispiel Linux

Interrupt-Vektor-Tabelle (2)

Die Interrupt-Vektor-Tabelle ist im System wie folgt definiert:

```
extern irq_desc_t irq_desc [NR_IRQS];
```

Aufbau eines Tabelleneintrags:

```
typedef struct {
        unsigned int status;            // IRQ-Status
        hw_irq_controller *handler;     // Zeiger auf verantwortlichen
                                        // Controller
        struct irqaction *action;       // Zeiger auf Action-Liste (Sharing von
IRQs)
        unsigned int depth;             // Spezielles Feld zum Aktivieren und
                                        // Deaktivieren des IRQ
} ____cacheline_aligned irq_desc_t;
```

Beispiel Linux

Linux, Action-Liste (3)

- action = Action-Descriptor, Struktur mit Verweis auf eigentliche ISR
- Verkettete Liste für jeden IRQ zum Zwecke des Interrupt-Sharings

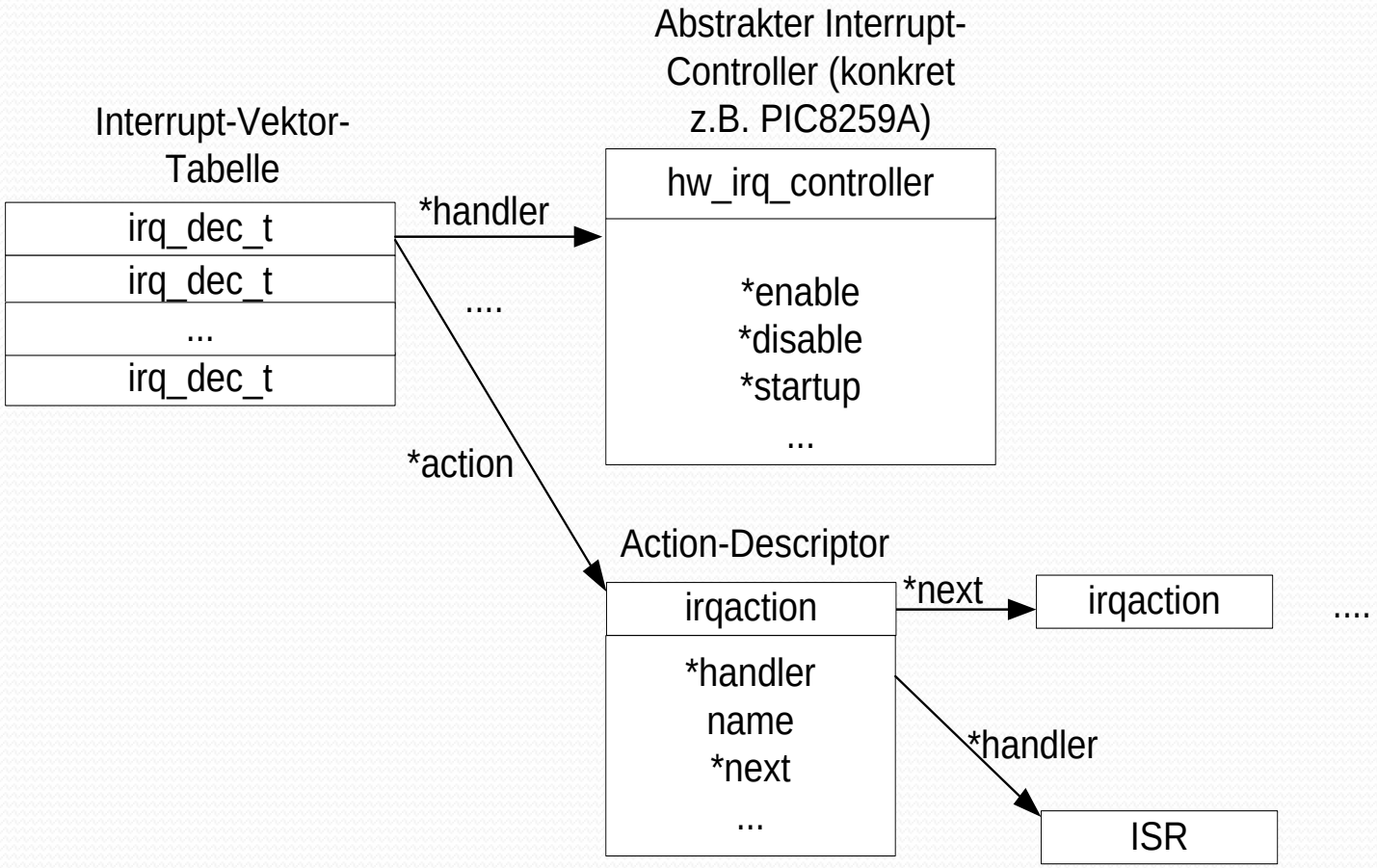
```
struct irqaction {

                                        // Verweis auf Interrupt-Service-
                                        // Routine
        void (*handler)(int, void *, struct pt_regs *);

        unsigned long flags;            // Eigenschaften des Interrupt-Handlers
        const char *name                // Name des Interrupt-Handlers
        void *dev_id;                   // Eindeutige Identifikation des
                                        // Interrupt-Handlers
        struct irqaction *next;         // Verweis auf nächsten Eintrag in der
                                        // Action-Liste
};
```

Beispiel Linux

Datenstrukturen im Kernel (4)

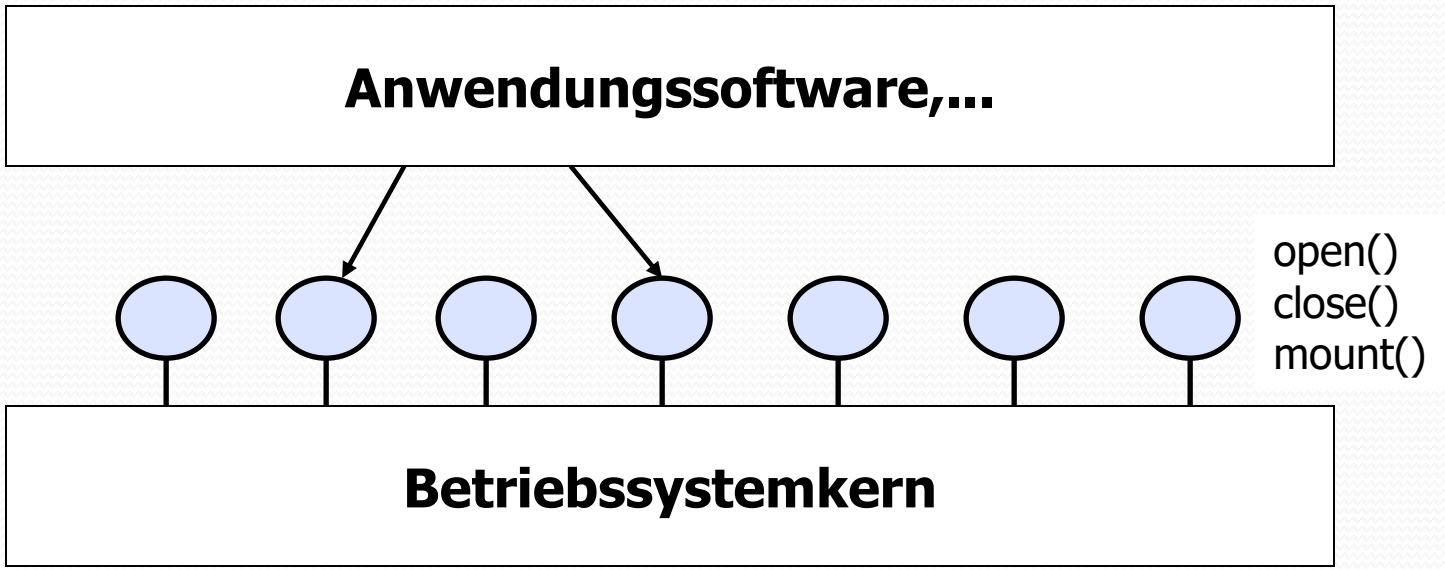


Dienste des Betriebssystems

- Anwendungsprogramme nutzen die **Dienste** des Betriebssystems, die über sog. Systemcalls aufgerufen werden
- Wohldefinierte Einsprungpunkte ins Betriebssystem
- Spezieller Aufrufmechanismus für einen Systemcall
  - **Software-Interrupt** (als **Trap** bezeichnet) oder **Supervisor Call** (SVC)
  - Vorteil: Anwendungsprogramm muss Adressen der Systemroutinen nicht kennen
- Alle Systemcalls zusammen bilden die Schnittstelle der Anwendungsprogramme zum Betriebssystemkern
  - Zugang zu Systemcalls wird meist in Bibliotheken bereitgestellt

Umschaltung in den Kernelmodus

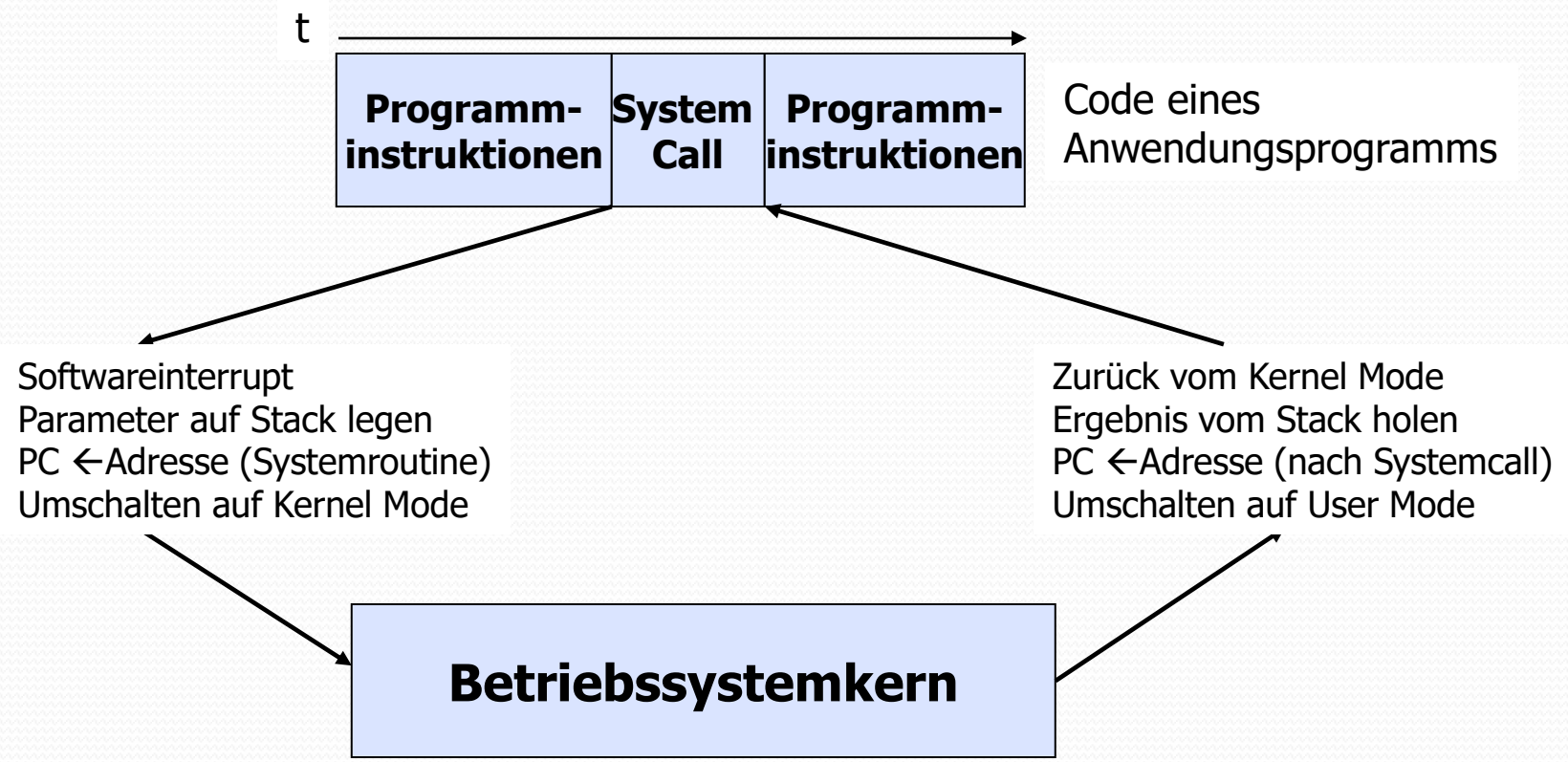
- Systemcalls werden im **Kernelmodus** ausgeführt
- Beim Aufruf wird durch den Prozessor vom Usermodus in den Kernelmodus **umgeschaltet**



Systemcall

Ablauf

- Befehlsfolge eines Systemaufrufs



Systemcalls bei POSIX

- Systemcalls sind standardisiert in IS 9945-1
- POSIX-Konformität erfüllen die meisten Unix-Derivate
- Beispiele:
  - fork(): Prozesserzeugung
  - execve(): Aufruf eines Programms
  - exit(): Beenden eines Prozesses
  - open(): Datei öffnen
  - close(): Datei schließen
  - read(): Daten aus Datei lesen
  - write(): Daten in Datei schreiben

POSIX = **P**ortable **O**perating **S**ystem Un**ix**

Systemcall-Ablauf

unter Linux/x86 Ablaufdiagramm

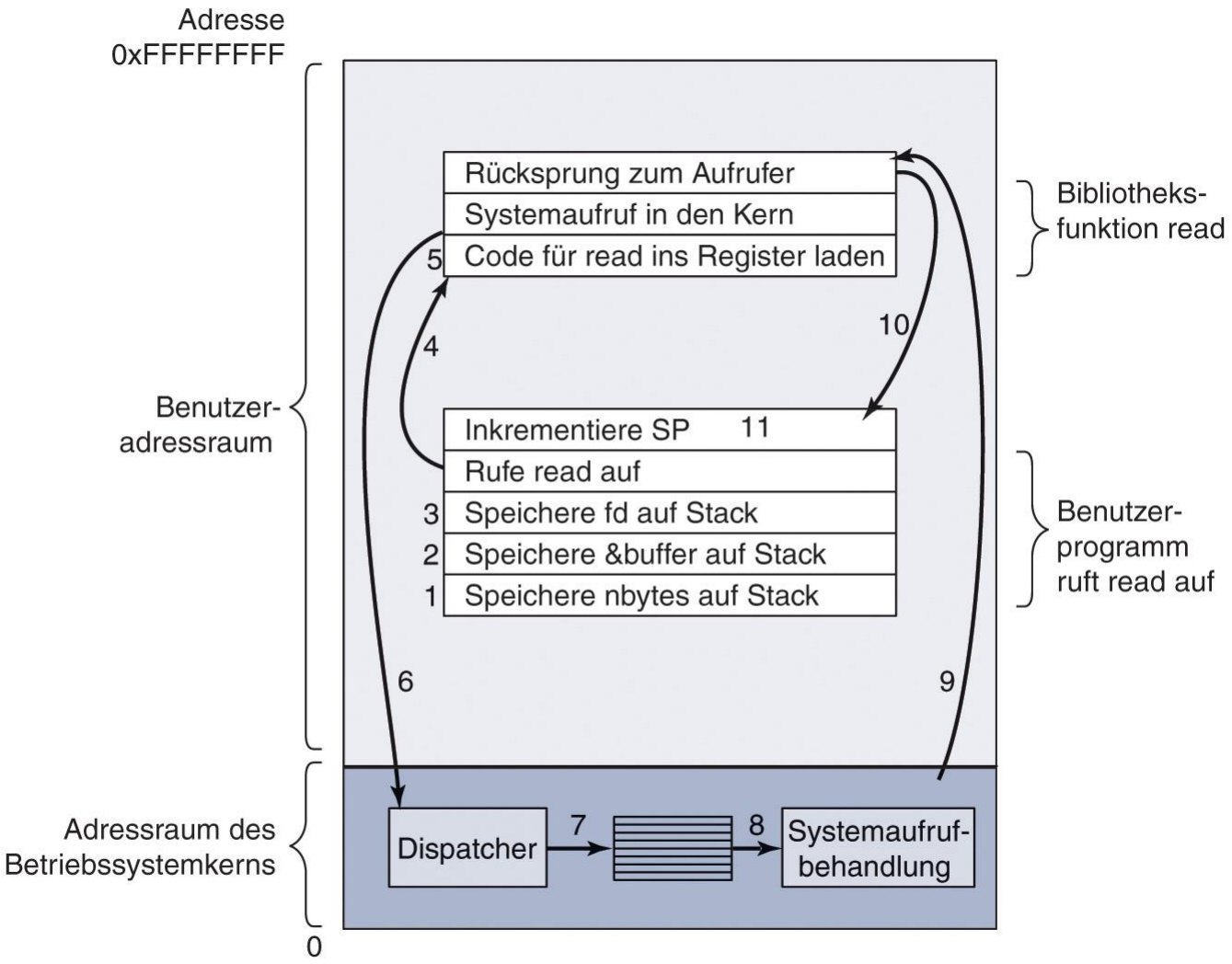


Abbildung 1.17: Die elf Schritte für den Systemaufruf `read(fd, buffer, nbytes)`

Aufgabe

- Erläutern Sie die wichtigsten Systemcalls des POSIX-Standards zur:
  - Processverwaltung
  - Dateiverwaltung
  - Verzeichnis- und Dateisystemverwaltung
  - Signale und Rechteverwaltung