Systeme I: Betriebssysteme

Kapitel 4 Prozesse

Maren Bennewitz

Version 20.11.2013

Begrüßung

- Heute ist Tag der offenen Tür
- Willkommen allen Schülerinnen und Schülern!

Wdhlg.: Attributinformationen in I-Nodes (urspr. Unix System V)

Feld	Bytes	Beschreibung
Mode	2	Dateityp, Schutzbits, setuid, setgid Bits
Nlinks	2	Anzahl der Verzeichniseinträge zu diesem I-Node (also max 2^{16})
Uid	2	UID des Besitzers
Gid	2	GID des Besitzers
Size	4	Dateigröße in Byte
Addr	39	Adresse der ersten zehn Blöcke, dann drei indirekte Blöcke
Gen	1	Generation (wird bei der Benutzung jedes Mal erhöht)
Atime	4	Zeitpunkt des letzten Zugriffs
Mtime	4	Zeitpunkt der letzten Änderung
Ctime	4	Zeitpunkt der letzten Änderung des I-Node

bei 3-Byte-Adressen

Nachtrag zu letzter Stunde

Felder im I-Node:

- Mtime (modify time): Zeit, zu der zuletzt der Inhalt der Datei verändert wurde
- Ctime (change time): Zeit, zu der zuletzt I-Node Informationen (z.B. Zugriffsrechte, Dateiname, etc.) verändert wurden

Inhalt Vorlesung

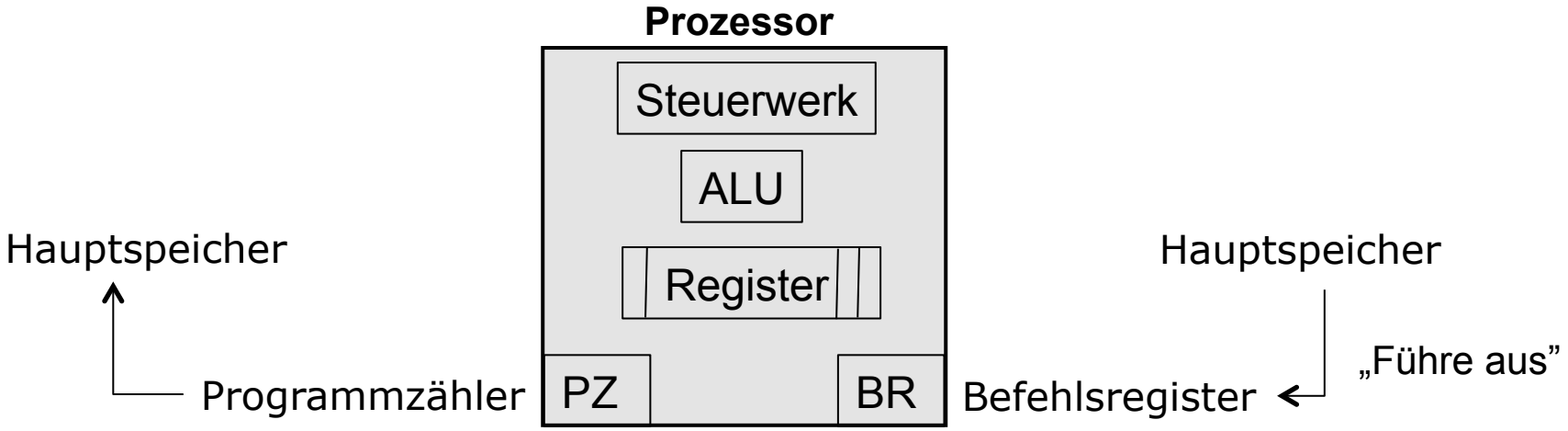
- Aufbau einfacher Rechner
- Überblick: Aufgabe, Historische Entwicklung, unterschiedliche Arten von Betriebssystemen
- Verschiedene Komponenten / Konzepte von Betriebssystemen
  - Dateisysteme
  - Prozesse
  - Nebenläufigkeit und wechselseitiger Ausschluss
  - Deadlocks
  - Scheduling
  - Speicherverwaltung

Einführung

- Aufgabe des Prozessors: Ausführen der Programme im Hauptspeicher
- Bei der Ausführung eines Programms wird ein Prozess erzeugt
- Die Befehle des Programms werden durch den Prozessor abgearbeitet

„Programm in Ausführung“

- Prozess = Instanz eines Programms mit
  - Aktuellem Wert vom Befehlszähler
  - Registerinhalten
  - Belegung von Variablen



[s. Kap. 1]

„Programm in Ausführung“

- Prozess = Instanz eines Programms mit
  - Aktuellem Wert vom Befehlszähler
  - Registerinhalten
  - Belegung von Variablen
- Multitasking-Betriebssysteme: Mehrere Prozesse können „pseudo-parallel“ (oder quasiparallel) ausgeführt werden

Pseudo-Parallelität

- Auf Prozessoren mit einem Kern laufen die Prozesse natürlich nicht wirklich parallel
- Sondern abwechselnd, wobei die Prozessorzuteilung durch das Betriebssystem geregelt ist
- Im Gegensatz zu echter Hardware-Parallelität bei Mehrkernprozessoren

Motivation Multitasking

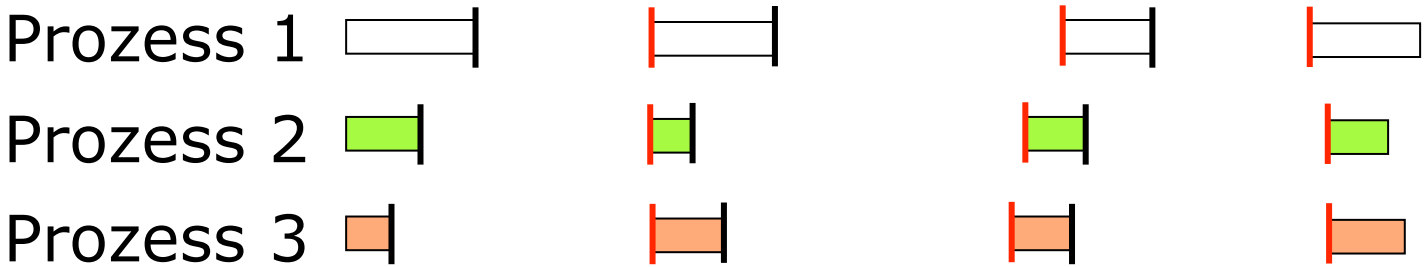
- Ein Benutzer will mehrere Aufgaben „gleichzeitig“ durchführen
- Mehrere Benutzer teilen sich einen leistungsfähigen Rechner (Timesharing)
- Die Durchführung einer einzigen Aufgabe lässt sich typischerweise in relativ unabhängige Teilaufgaben zerlegen
- Ein getrennter Prozess für jede Aufgabe
- Pseudo-parallele Ausführung mehrerer Prozesse

Warum kann Multitasking überhaupt funktionieren?

- Rechner sind so leistungsfähig, dass die pseudo-parallele Ausführung mehrerer Prozesse schnell genug abläuft
- Viele Prozesse können den Prozessor ohnehin nicht permanent nutzen
- Grund: Viel Zeit wird mit Warten auf Ein-/ Ausgaben verbracht

Beispiel

- Getrennte Ausführung



- Pseudo-parallele Ausführung

Zerlegung in Teilaufgaben

- Durch Zerlegung möglich: Aufteilung der Rechenzeit unter verschiedenen Teilaufgaben durch das Betriebssystem („Scheduling“)
- Wartezeiten auf Ein-/Ausgaben werden automatisch durch andere Prozesse genutzt
- Ein-Programm-Lösung mit gleicher Funktionalität hätte häufig verworrene Kontrollstruktur („Spaghetti-Code“)

Adressraum

- Zu jedem Prozess gehört ein Adressraum im Hauptspeicher
- Liste von Speicherzellen mit Adressen, in denen der Prozess lesen und schreiben darf
- Adressraum enthält
  - Ausführbares Programm
  - Programmdaten
  - Kellerspeicher (“Stack”, für lokale Variablen)

Prozessinformationen

- Individuelle Prozessinformationen von Prozessorregistern:
  - Programmzähler
  - Allgemeine Register
  - Stack pointer (zeigt auf oberstes Kellerelement)
- Prozess ID, Priorität, Zustand, geöffnete Dateien, Startzeit, etc.
- Diese Informationen sind in einer sog. Prozesstabelle gespeichert („activation record")

Prozesswechsel (1)

- Prozesswechsel („Context Switch“): Wechsel von der Ausführung eines Prozesses zu der Ausführung eines anderen
- Dispatcher: Teil des Betriebssystems, der Prozesswechsel durchführt
- Scheduler: Teil des Betriebssystems, der den Prozessen Rechenzeit auf dem Prozessor zuweist

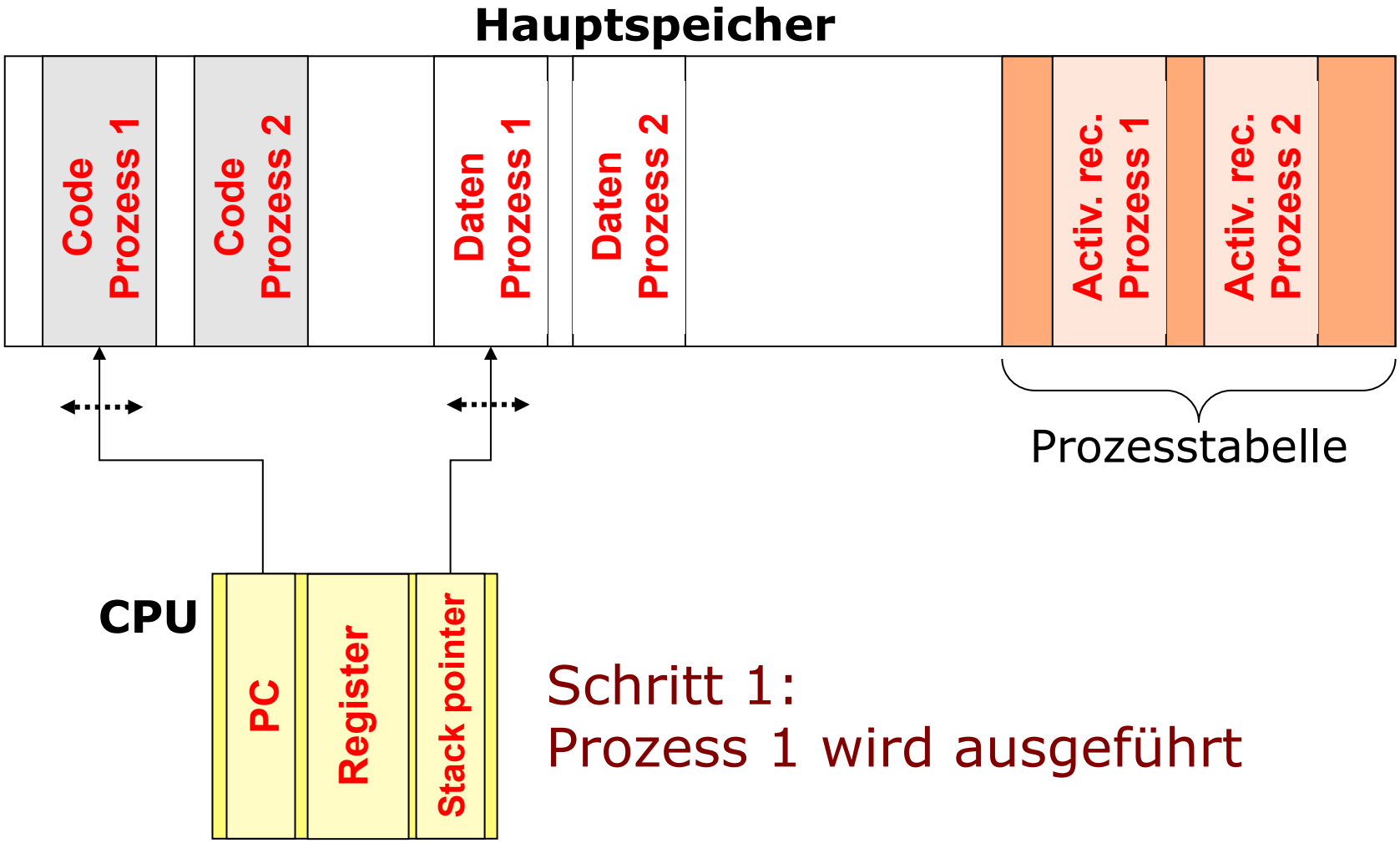
Prozesswechsel (2)

- Nicht-präemptive Betriebssysteme, z.B. MS-DOS
- Prozessen kann nur dann der Prozessor entzogen werden, wenn sie ihn selbst abgeben wegen
  - Terminierung
  - Warten auf Abschluss einer Ein- / Ausgabeoperation
  - Warten auf Zuteilung einer Ressource

Prozesswechsel (3)

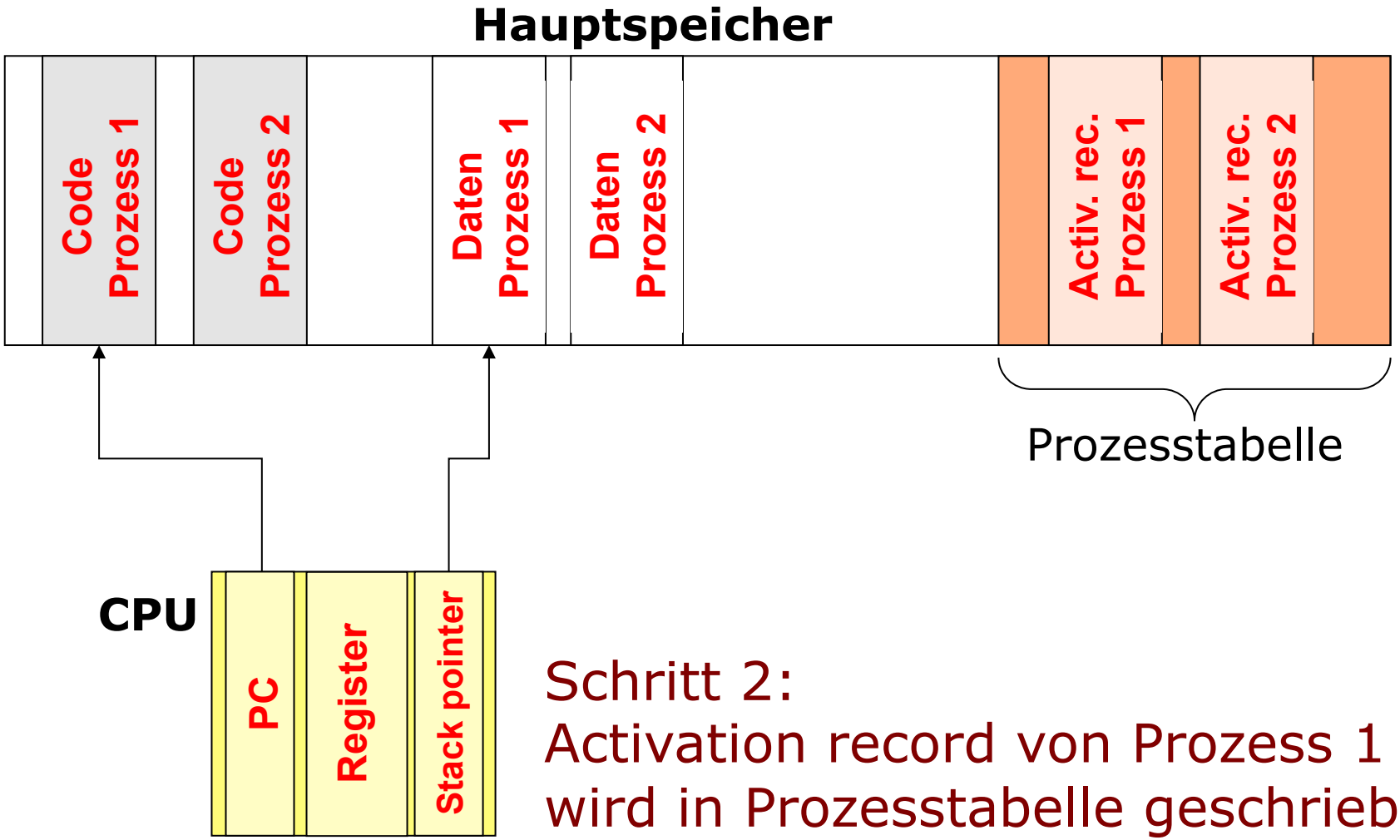
- Präemptive Betriebssysteme, z.B. Unix, Linux, Windows
- Der aktive Prozess kann unterbrochen werden vom Betriebssystem (z.B. wenn ein neuer Prozess gestartet wurde)
- Oder auch: Neuzuteilung des Prozessors wird in regelmäßigen Zeitintervallen vom Betriebssystem erzwungen
- Mehr Verwaltungsaufwand, aber bessere Auslastung der CPU

Beispiel: Prozesswechsel



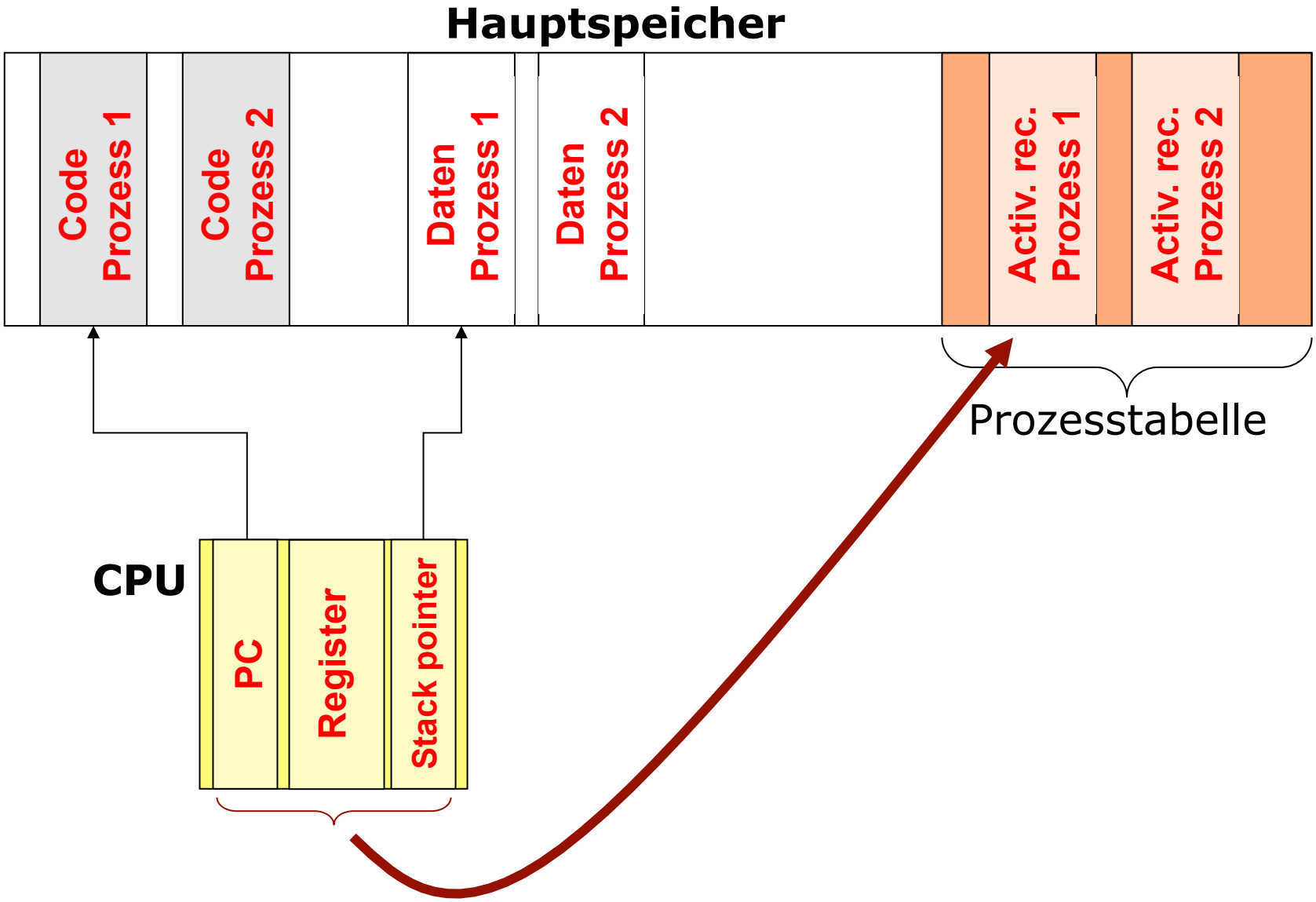
Schritt 1:
Prozess 1 wird ausgeführt

Beispiel: Prozesswechsel

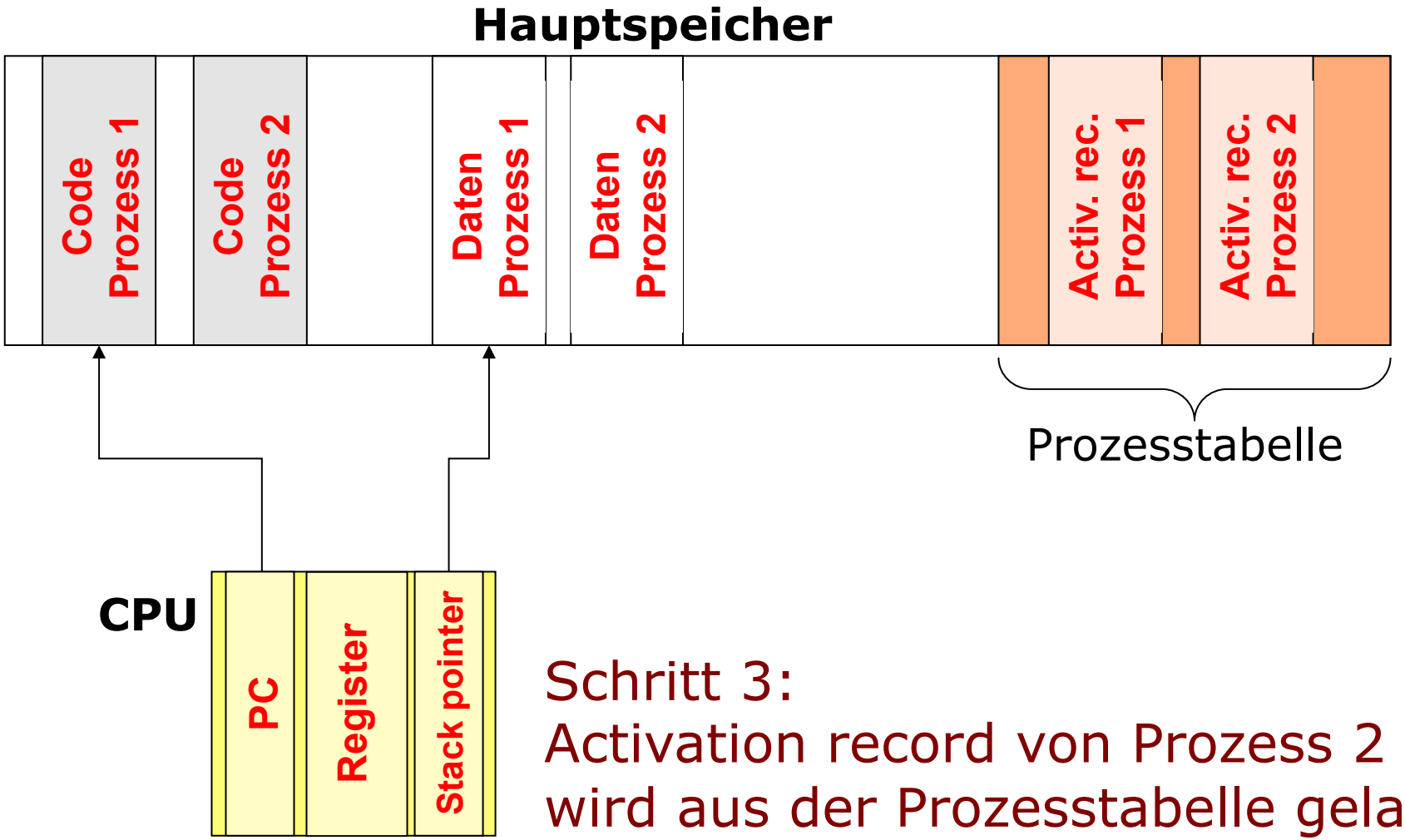


Schritt 2:
Activation record von Prozess 1 wird in Prozesstabelle geschrieben

Beispiel: Prozesswechsel

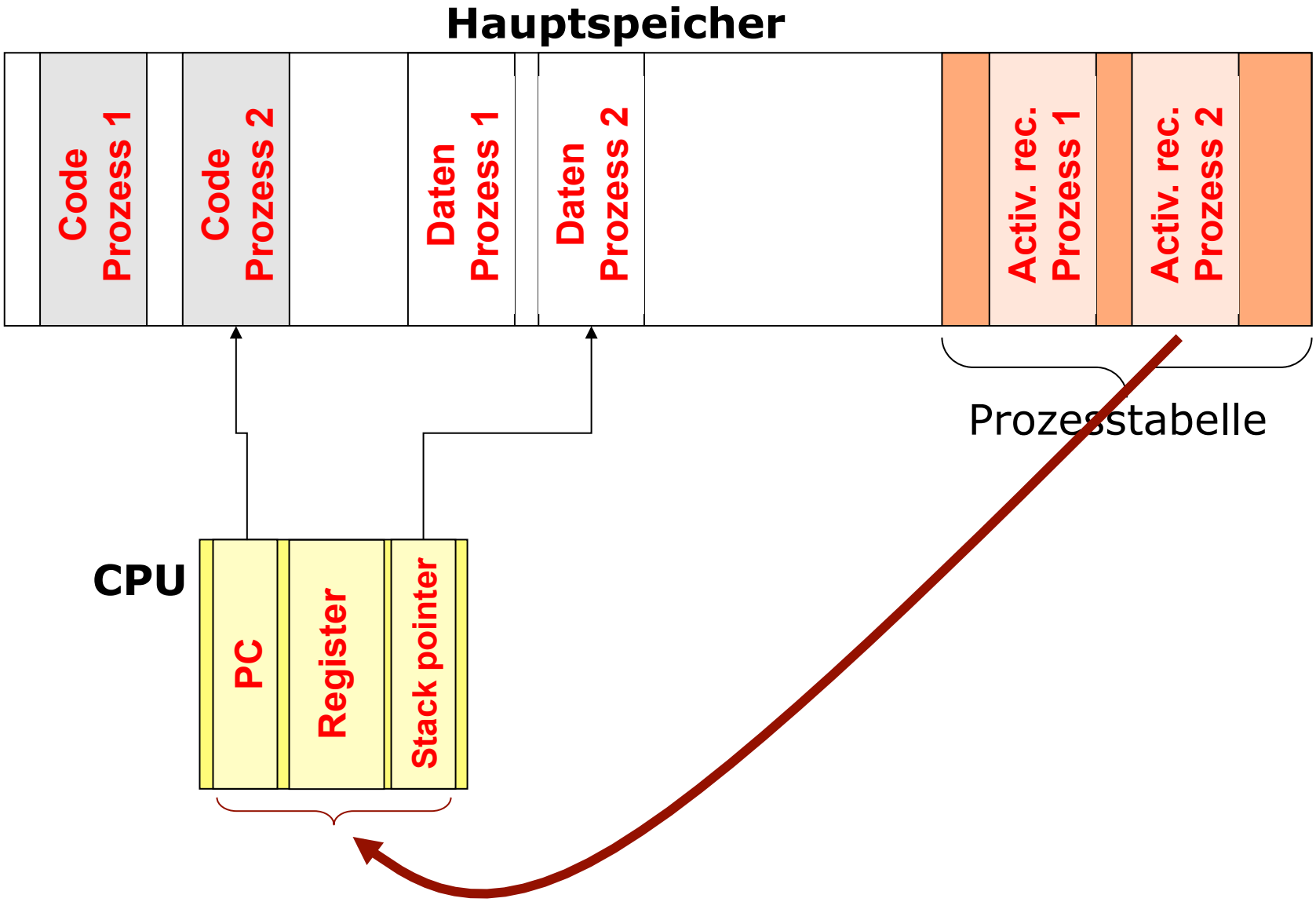


Beispiel: Prozesswechsel

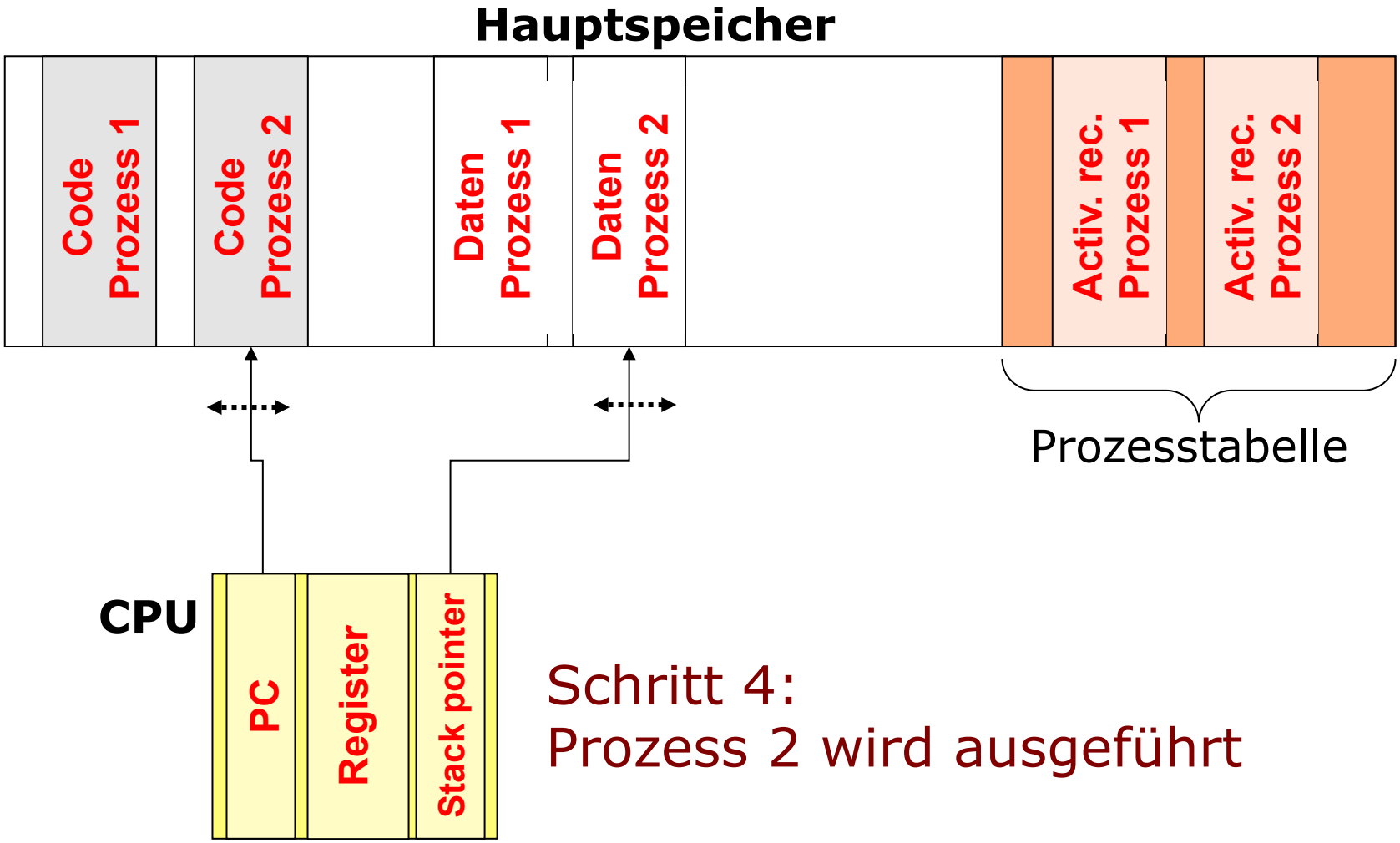


Schritt 3:
Activation record von Prozess 2 wird aus der Prozesstabelle geladen

Beispiel: Prozesswechsel



Beispiel: Prozesswechsel



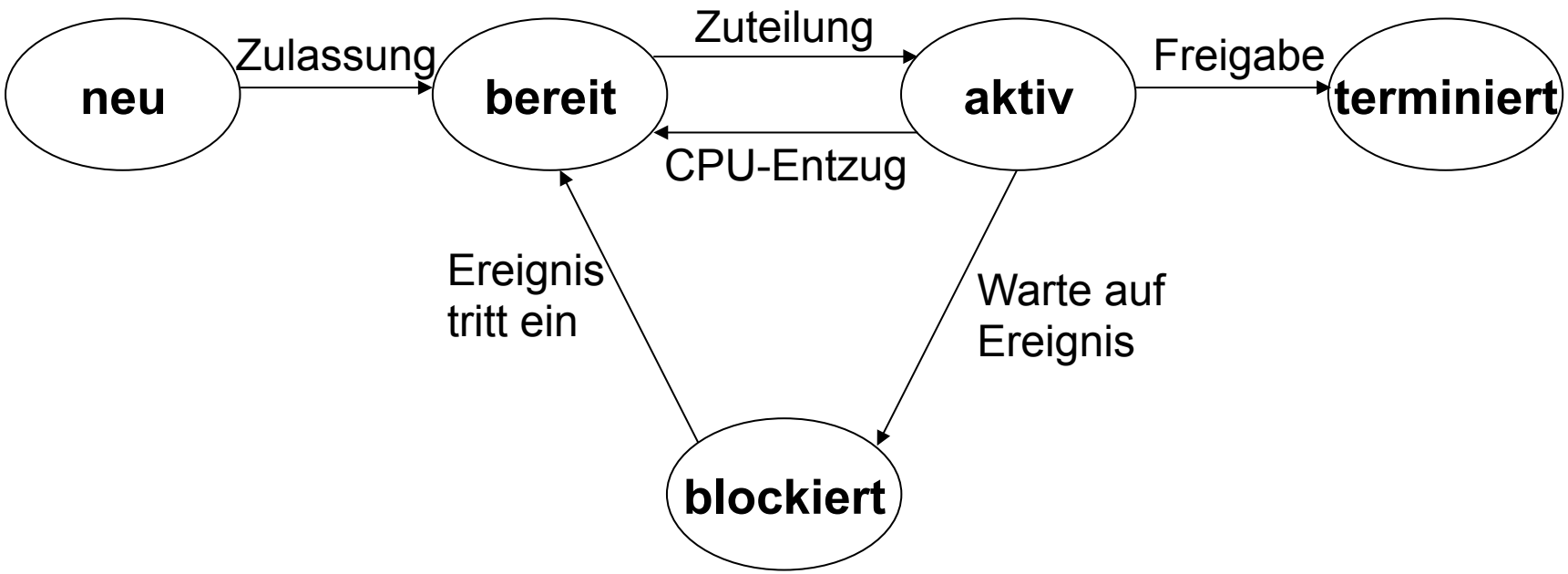
Schritt 4:
Prozess 2 wird ausgeführt

Prozesszustände

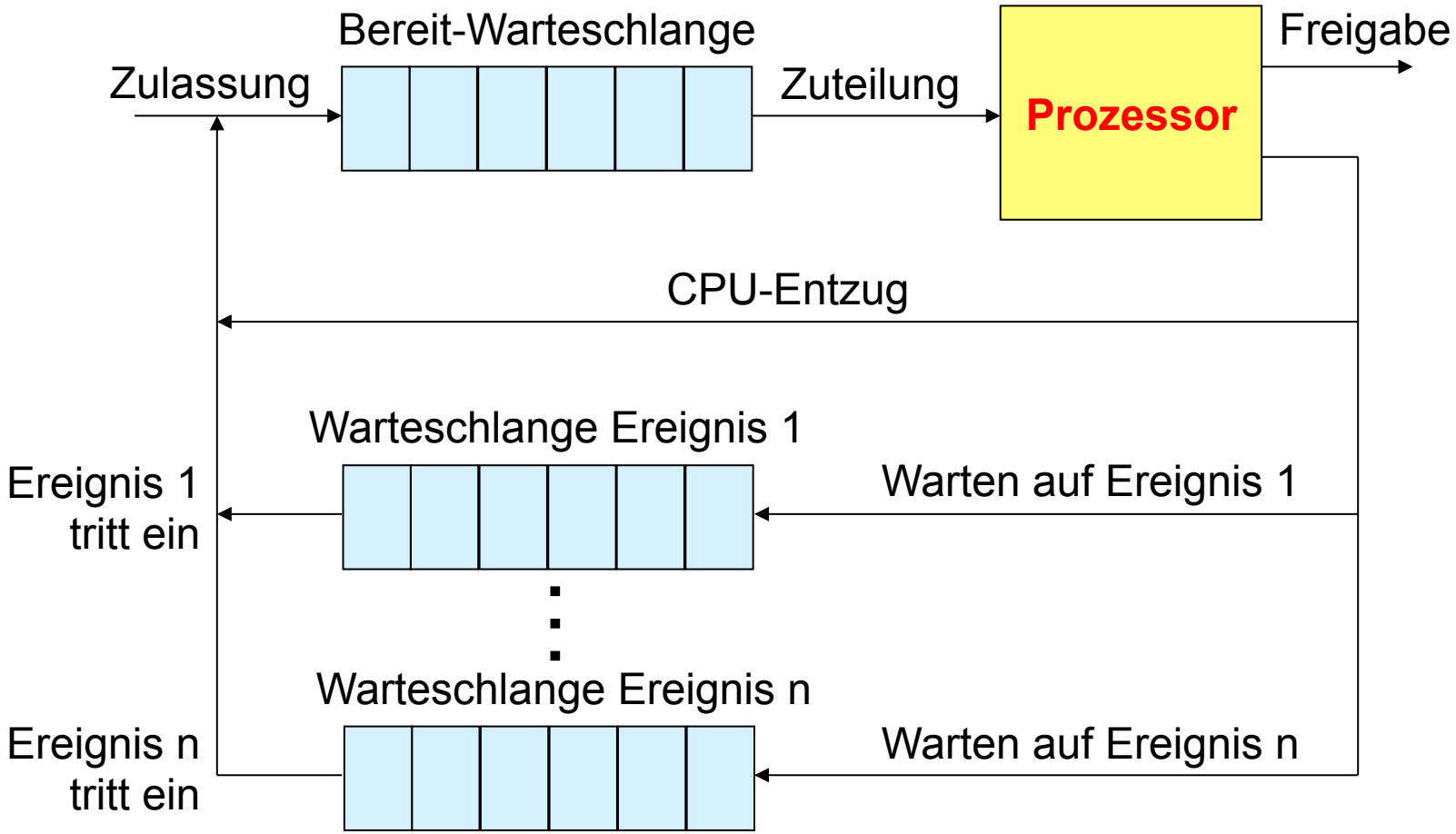
Modell mit 5 Zuständen:

- **Neu**: Prozess wurde erzeugt, ist aber noch nicht gestartet
- **Bereit**: Rechenbereit, aber Prozessor ist diesem Prozess nicht zugeteilt
- **Aktiv**: CPU ist dem Prozess zugeteilt
- **Blockiert**: Nicht in der Lage weiterzuarbeiten, wartet auf etwas (z.B. E/A)
- **Terminiert**

Prozesszustände



Warteschlangen von Prozessen, die bereit oder blockiert sind



Auslagern von Prozessen

- Für bessere Auslastung:
  - Der Prozessor kann sich trotz Multitasking die meiste Zeit im Leerlauf befinden
  - Eine Möglichkeit: Hauptspeicher ausbauen, um mehr Prozesse aufzunehmen
  - Besser: Verschieben von Prozessen auf Festplatte, dadurch Schaffen von Freiraum
- Oder z.B. auch wenn Prozess mit höherer Priorität bereit ist, ausgeführt zu werden, oder bei periodischen Prozessen, wenn der Hauptspeicher belegt ist

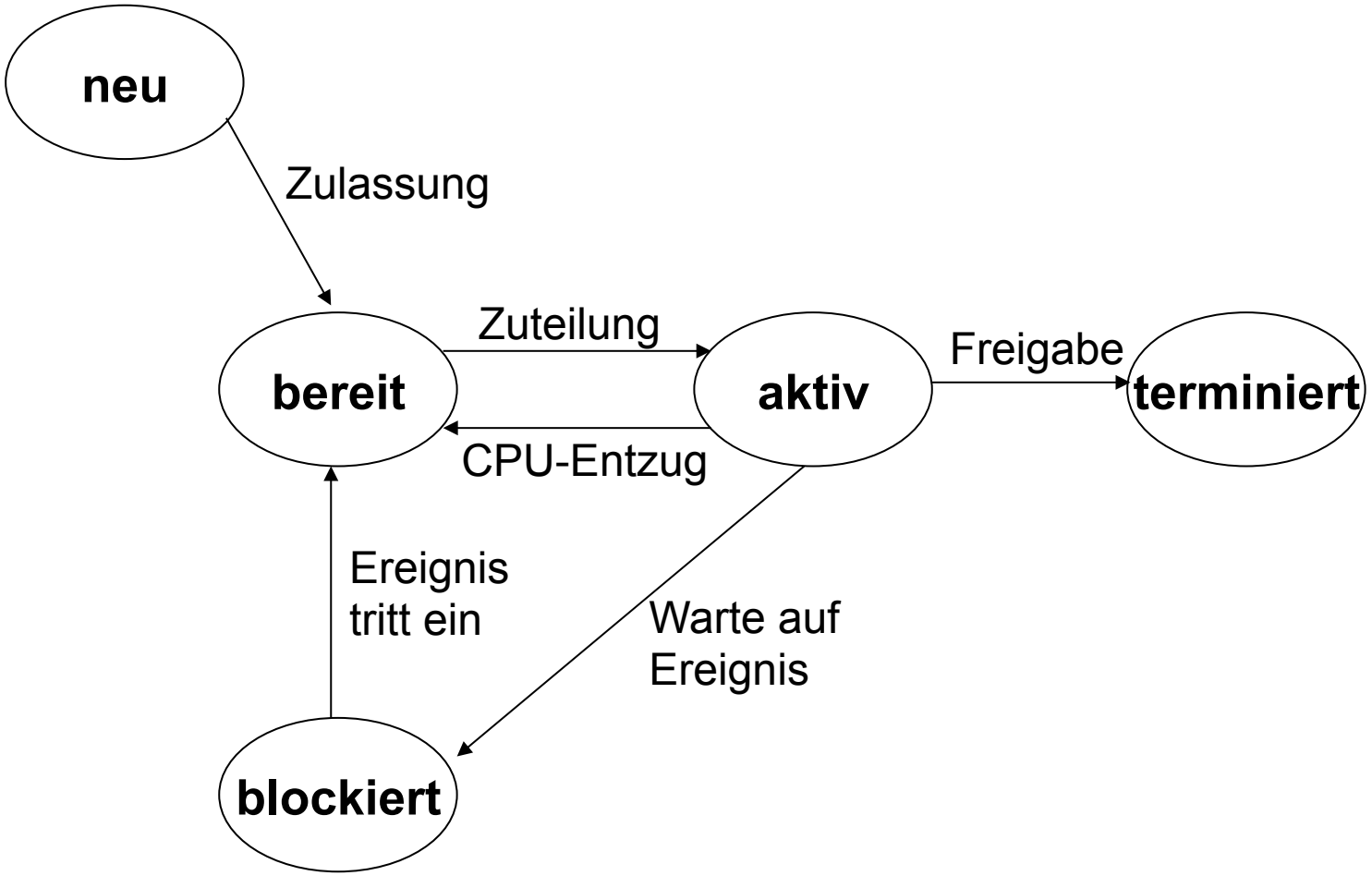
Swapping (Auslagern)

- Prozesse werden aus dem Hauptspeicher entfernt
- Daten von bereiten oder auf ein Ereignis wartenden Prozessen werden auf die Festplatte ausgelagert
- Beachte: Swapping verursacht Kosten (Laufzeit)
- Neue Prozesszustände nötig

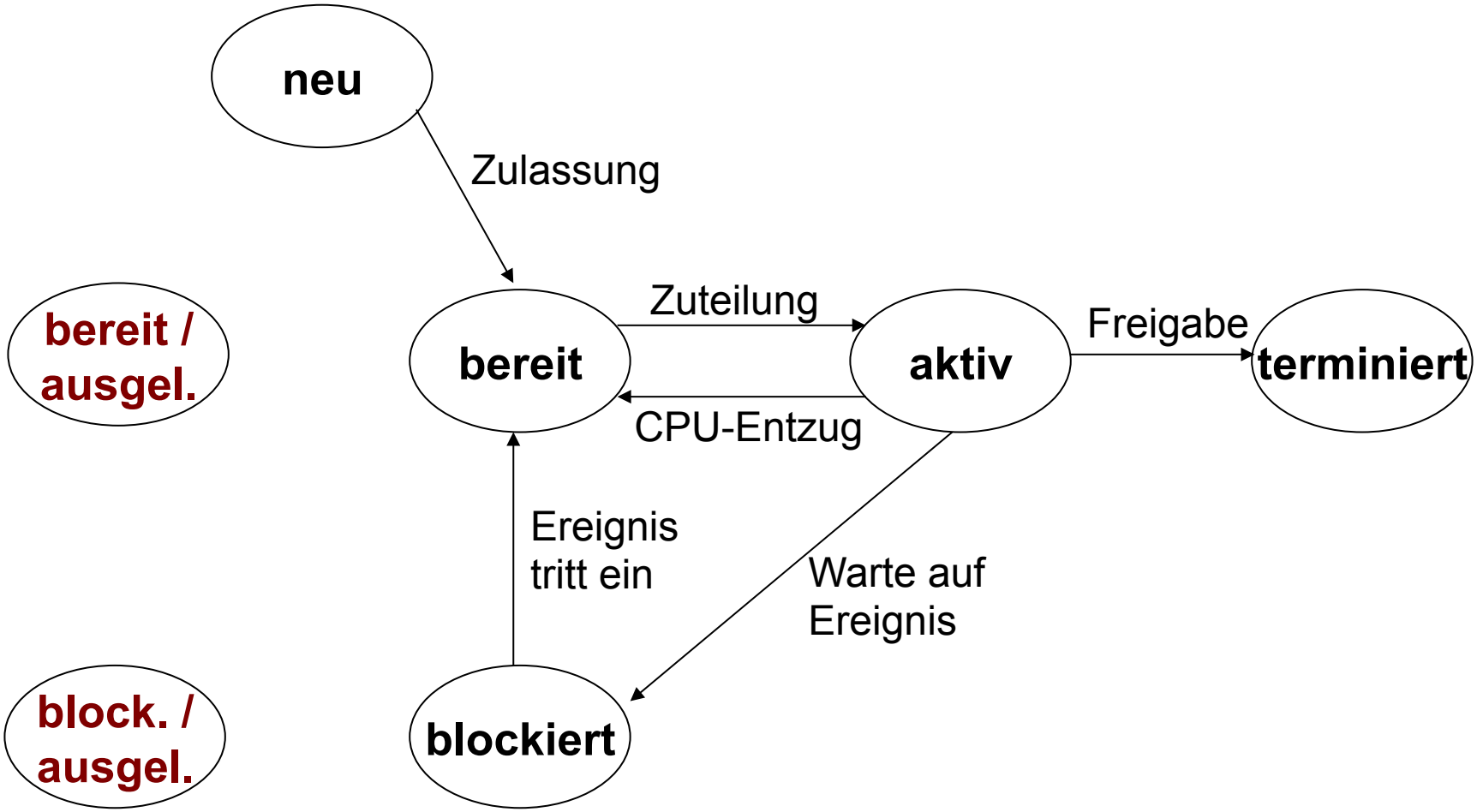
Neue Zustände für Swapping

- Unterscheide, ob ein Prozess
  - auf ein Ereignis wartet oder nicht und
  - ob er ausgelagert wurde oder nicht
- Dafür sind vier Zustände nötig:
  - Bereit: Im Hauptspeicher, rechenbereit
  - Blockiert: Im Hauptspeicher, wartet auf Ereignis
  - Blockiert und ausgelagert: Auf Festplatte, wartet auf Ereignis
  - Bereit und ausgelagert: Auf Festplatte, aber rechenbereit

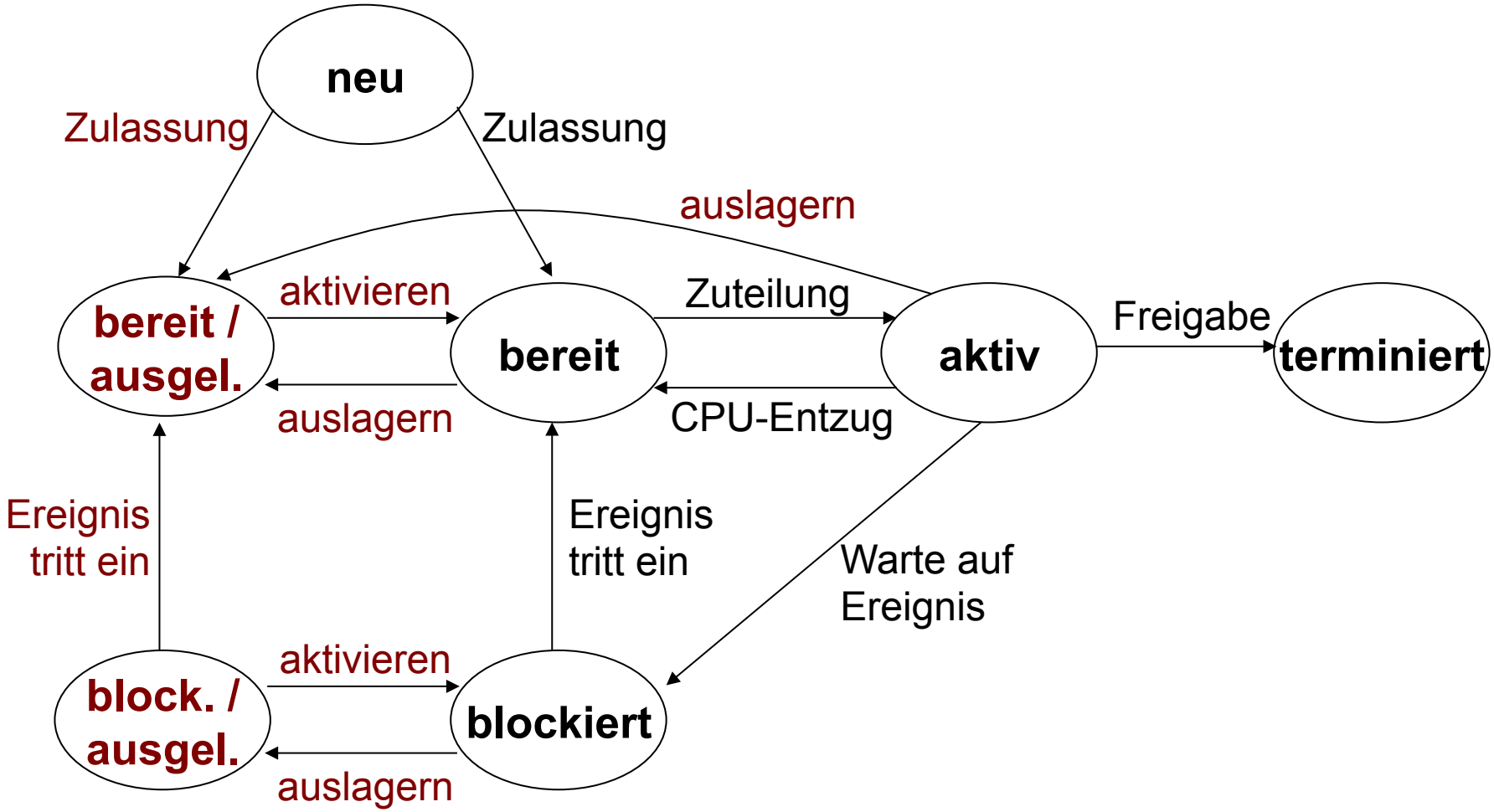
Prozesszustände



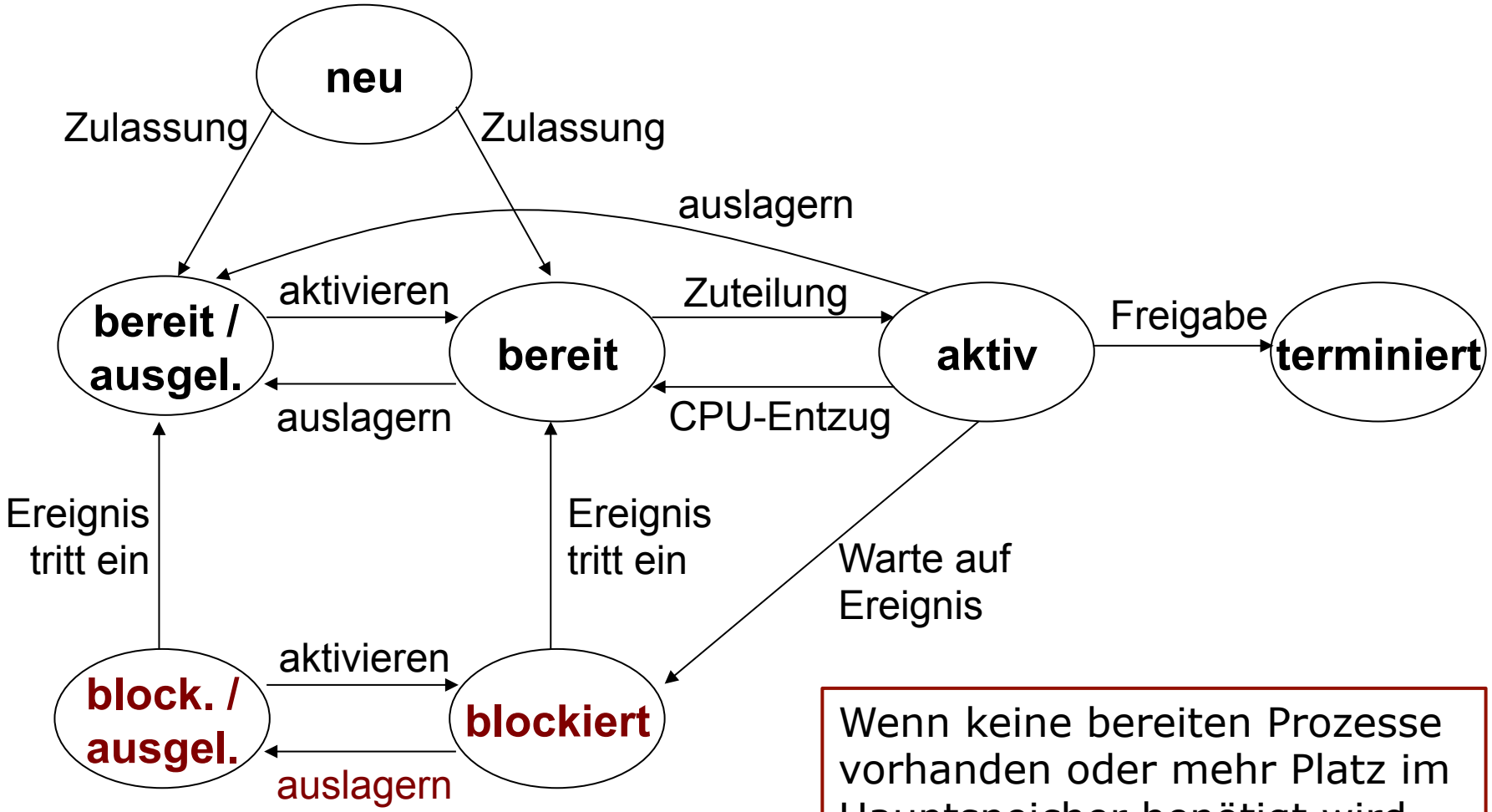
Prozesszustände



Prozesszustände

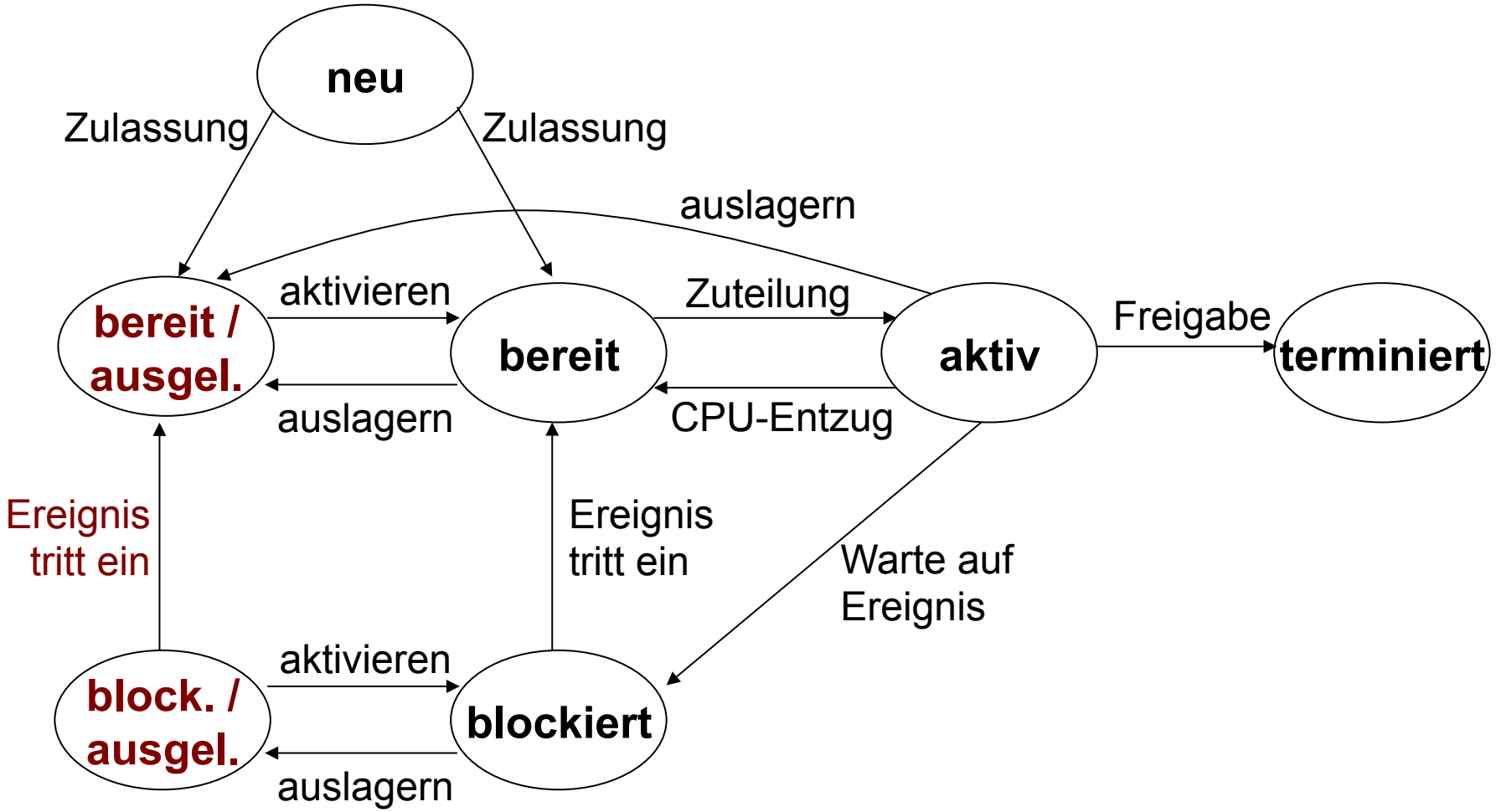


Prozesszustände

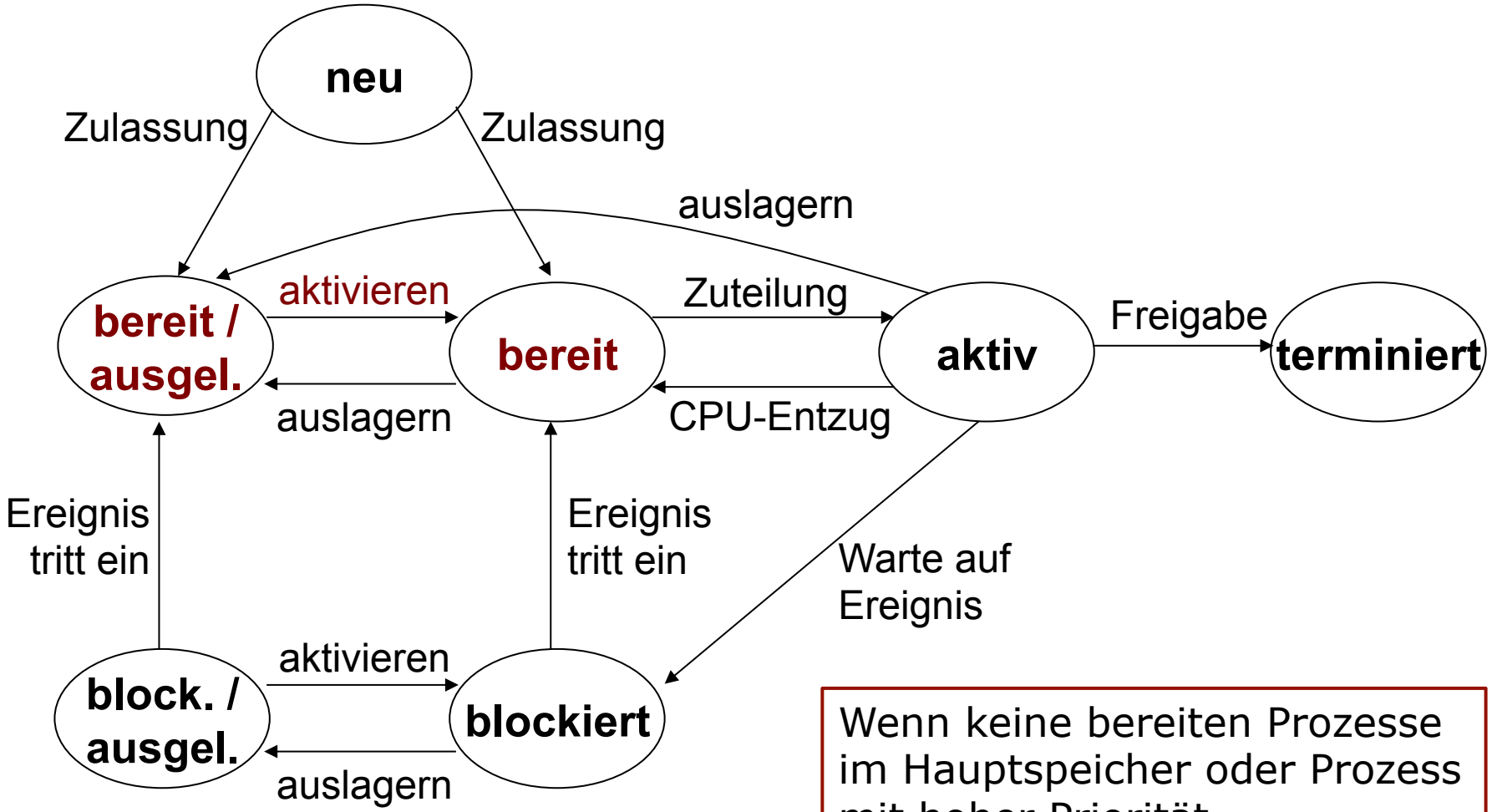


Wenn keine bereiten Prozesse vorhanden oder mehr Platz im Hauptspeicher benötigt wird

Prozesszustände

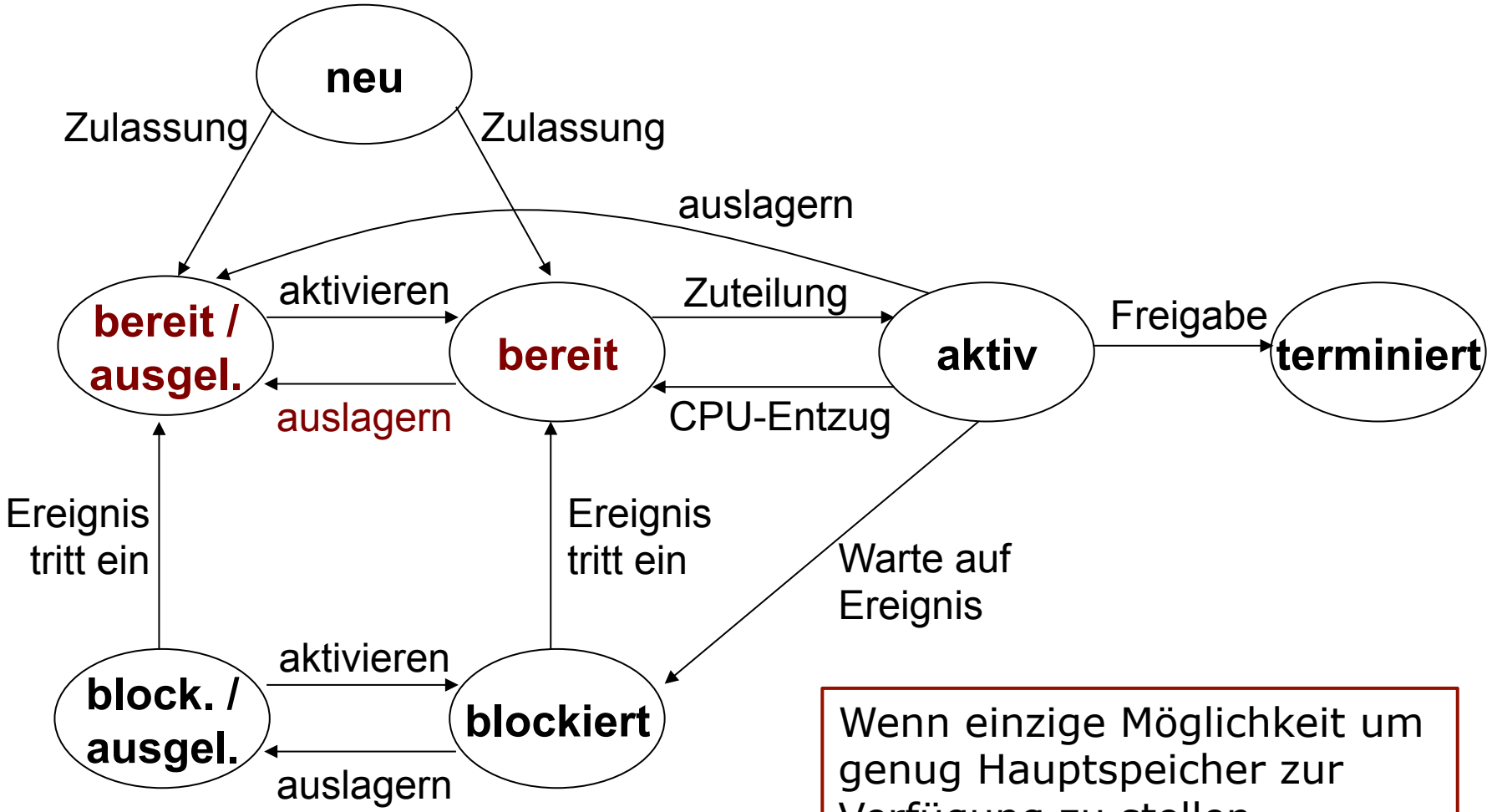


Prozesszustände



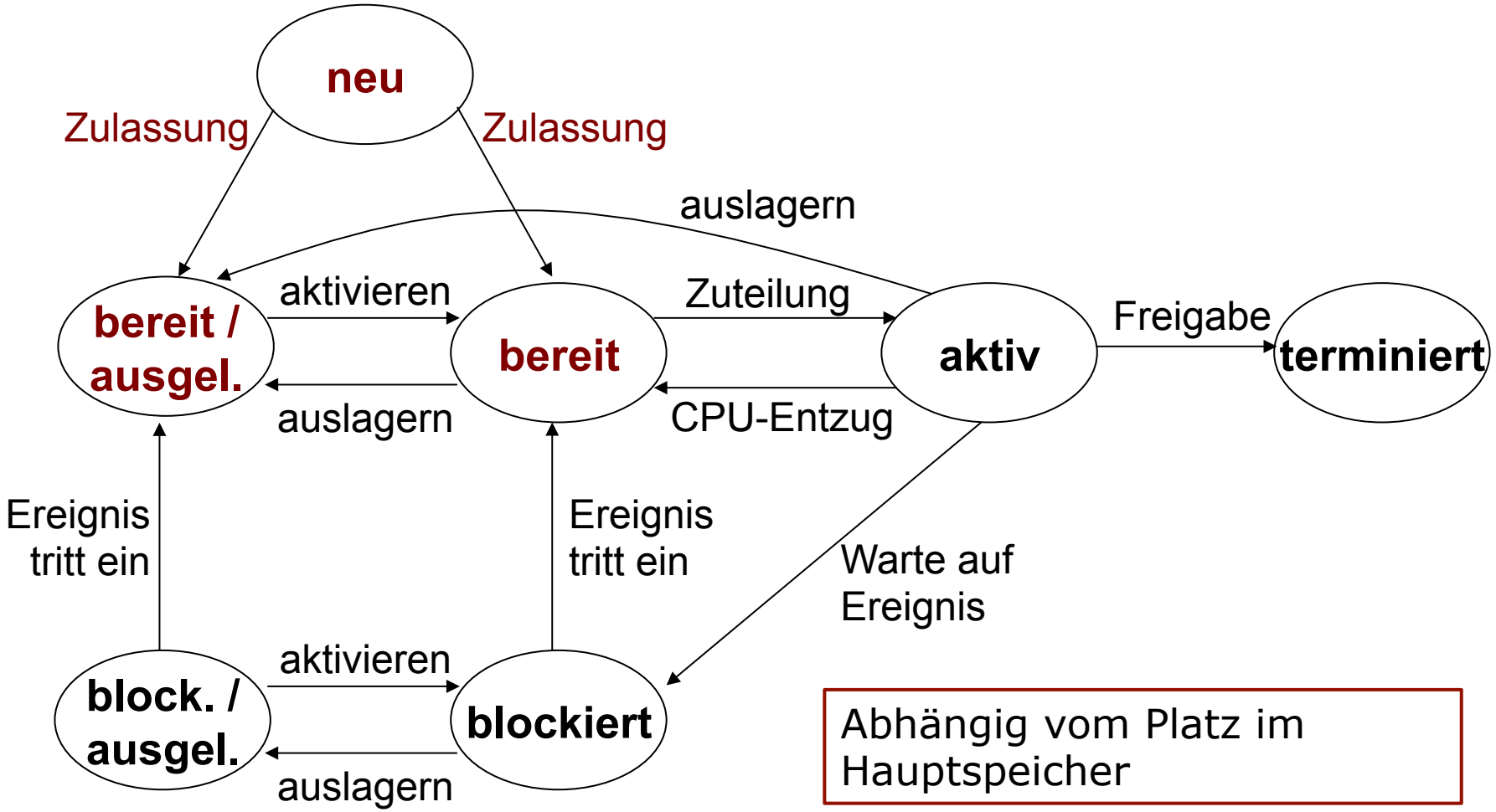
Wenn keine bereiten Prozesse im Hauptspeicher oder Prozess mit hoher Priorität

Prozesszustände

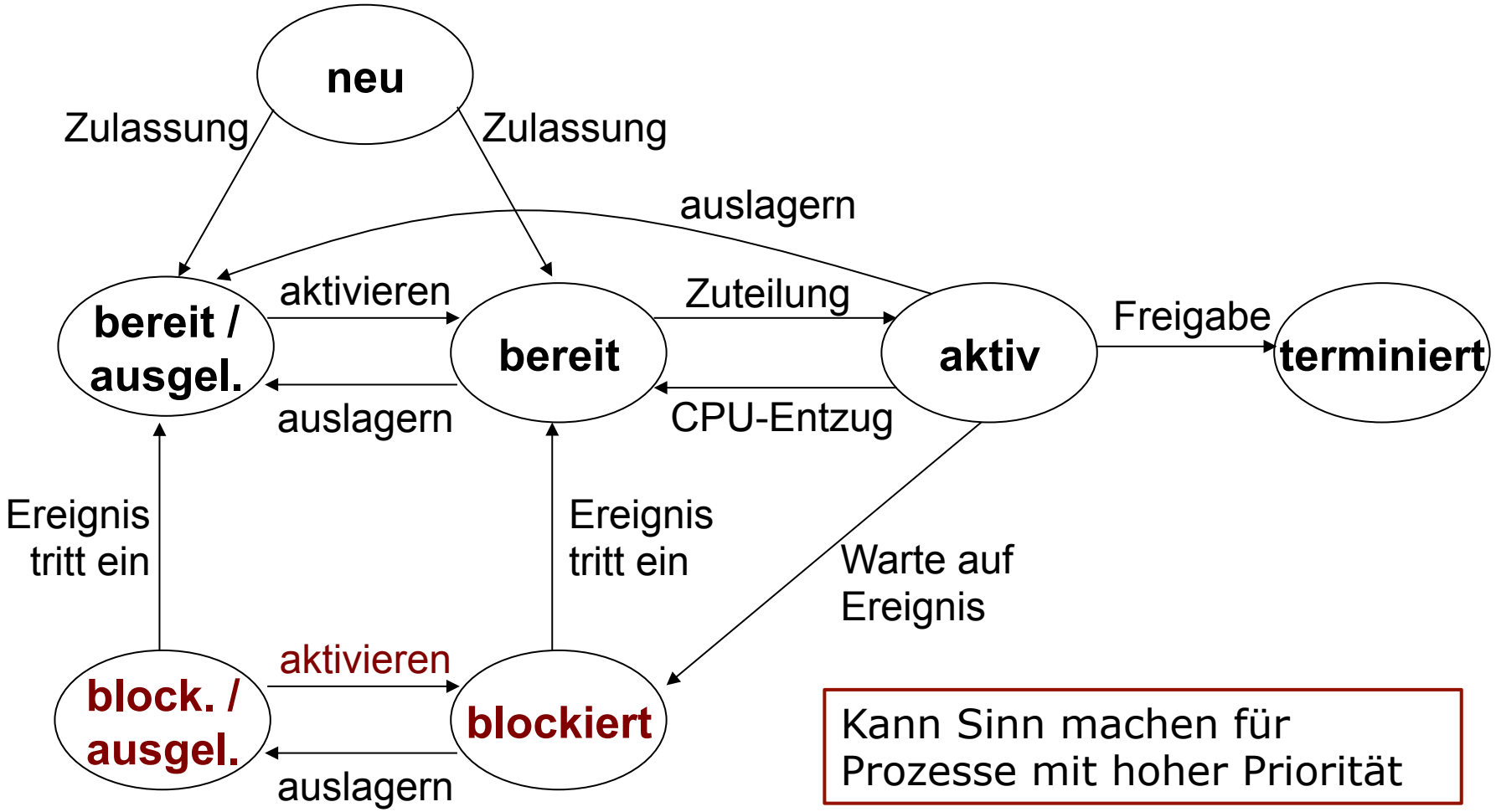


Wenn einzige Möglichkeit um genug Hauptspeicher zur Verfügung zu stellen

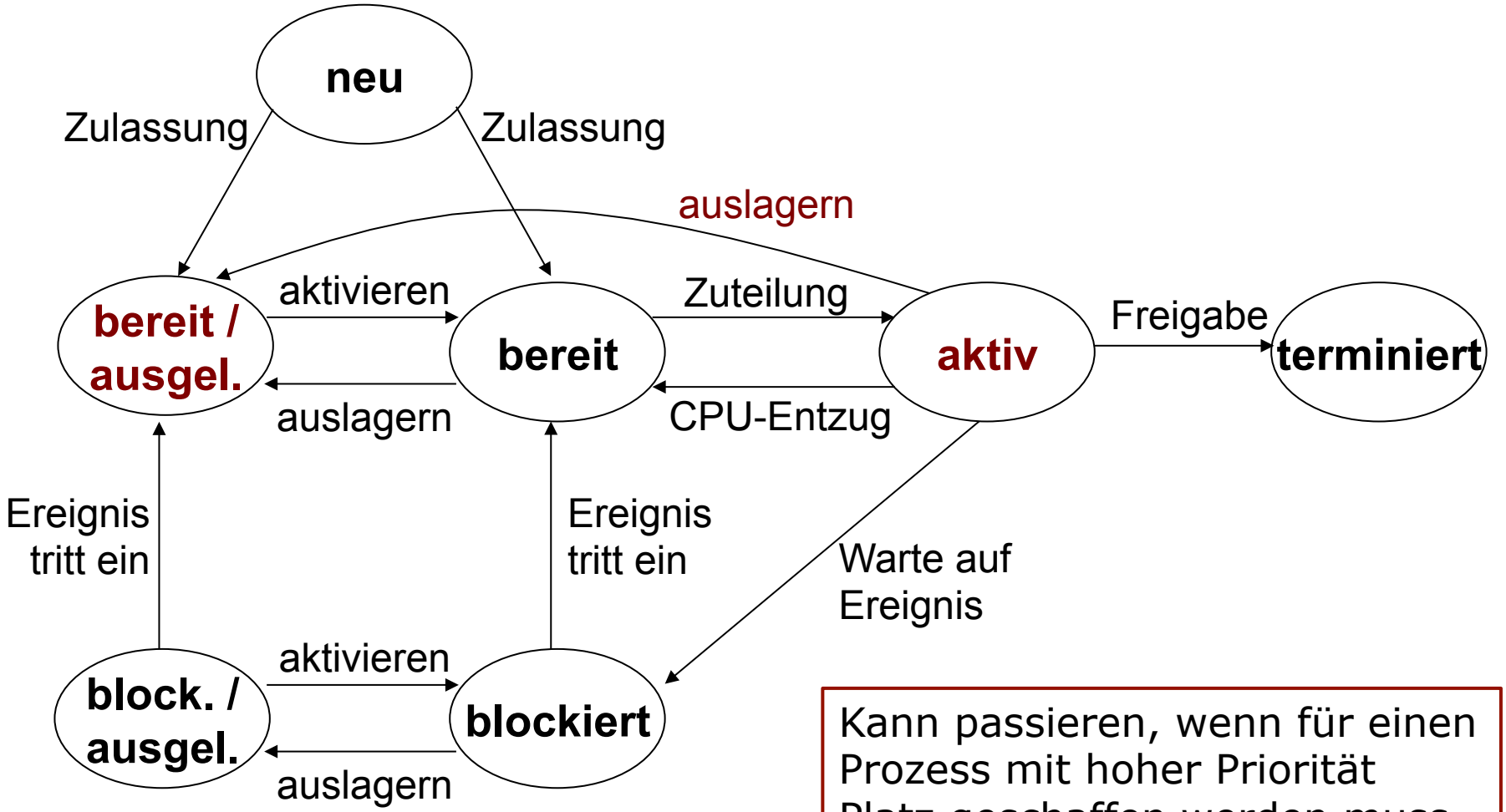
Prozesszustände



Prozesszustände



Prozesszustände



Kann passieren, wenn für einen Prozess mit hoher Priorität Platz geschaffen werden muss

Hinweise

- Programme können auch laufen, wenn sich nur ein Teil von ihnen im Hauptspeicher befindet
- Virtueller Speicher eines Prozesses: Sein Hauptspeicherbereich + Bereiche in einer Auslagerungsdatei auf der Festplatte
- Virtuelle Adresse: Ein Ort im Speicher
- Betriebssystem verwaltet Speicher und Zugriff (Paging, Kapitel 8)
- Scheduling: Zuweisung von CPU-Zeit (Kapitel 7)

Interprozesskommunikation

- Adressräume verschiedener Prozesse sind
  - Getrennt voneinander
  - Geschützt gegen den Zugriff anderer Prozesse
- Kommunikation durch
  - „Shared Memory“: Gemeinsam genutzte Arbeitsspeicherbereiche (schreiben, lesen)
  - Betriebssystemfunktionen zum Senden und Empfangen von Nachrichten (Kommunikation über Adressraum des Betriebssystems)

Threads (1)

- Mini-Prozesse („leichtgewichtige Prozesse“)
- Threads haben eigenen Befehlszähler, Register, Stack, Zustand
- Zustände: Aktiv, blockiert, bereit
- Mehrere Threads können parallel in einem Prozess laufen („Multithreading“)
- Prozessorkerne wechseln schnell zwischen Threads hin und her

Threads (2)

- Besitzen gemeinsamen Adressraum
- Können sich globale Variablen, geöffnete Dateien etc. teilen
- Kein Schutz voreinander
- Annahme: Threads kooperieren untereinander und teilen sich Ressourcen
- Weniger Verwaltungsaufwand im Vergleich zu Prozessen; schnellere Erstellung
- Beispiel: Textverarbeitung mit Benutzer-interaktion und Backup

Zusammenfassung

- Prozess = “Programm in Ausführung”
- Alle zu einem Prozess gehörigen Daten werden im Hauptspeicher verwaltet
- Prozesse konkurrieren um Rechenzeit
- Betriebssystem weist den Prozessen Rechenzeit zu und führt Prozesswechsel durch
- Der Hauptspeicher reicht nur für begrenzte Anzahl von Prozessen
- Es kann nötig sein, Prozesse aus dem Hauptspeicher temporär auszulagern