2. Verarbeitungsgrundlagen

2.1 Multiprogrammierung

2.1.1 Rechnerstruktur

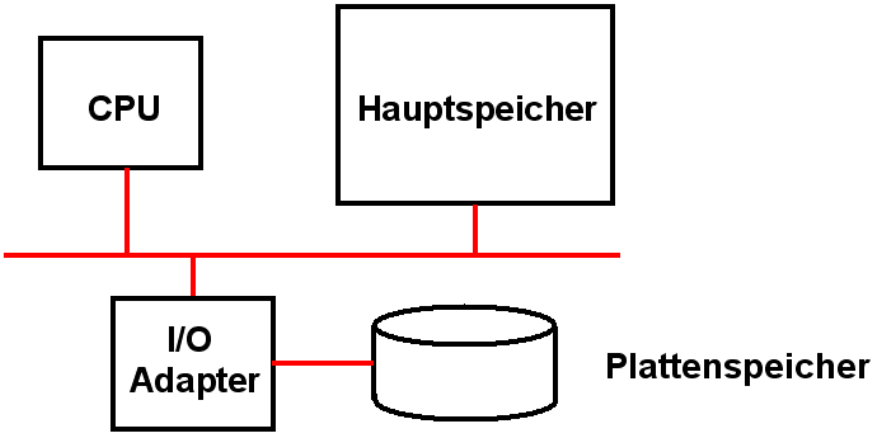

Abb. 2.1.1
Struktur eines Rechners

Ein moderner Rechner besteht aus einer CPU, welche ein Programm ausführt, einem Hauptspeicher (Main Storage, oft auch Arbeitspeicher genannt), welcher das auszuführende Programm und temporäre Daten enthält, und einem Anschluss (I/O Adapter) für einen (oder viele) Plattenspeicher, auf dem ausführbare Programme in Bibliotheken (Libraries) und Daten in Dateien und Datenbanken gespeichert sind. Der Hauptspeicher wird in Silizium Technologie implementiert, in der Form sog. DIMMs.

Ein auszuführendes Programm besteht aus einer Folge von Maschinenbefehlen (Machine Instructions). Das Format der Maschinenbefehle wird durch die Rechnerarchitektur bestimmt. Architekturen wie x86 (Pentium), PowerPC, Sparc, Itanium und System z (Mainframe) haben unterschiedliche Formate der Maschinenbefehle.

Ein Anwendungsprogramm, auch Benutzerprogramm genannt (application program, user program) wird heutzutage fast immer in einer höheren Programmiersprache wie Java, C++, Cobol, PL/1, ADA geschrieben, und durch einen Compiler (oder Interpreter) in ein Maschinensprache-Programm übersetzt, welches aus einer Folge von Maschinenbefehlen besteht. Programme in einer höheren Programmiersprache können nie direkt ausgeführt werden, sondern müssen vorher übersetzt (compiled oder interpreted) werden.

2.1.2 Quellprogramm und Maschinenprogramm

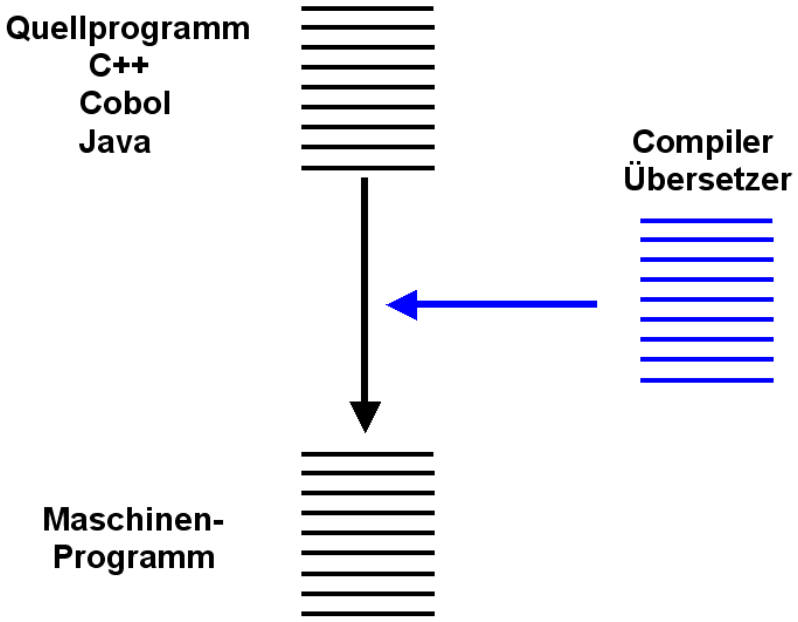

Abb. 2.1.2
Programm Übersetzung

Benutzerprogramme (andere Bezeichnung Anwendungsprogramm, engl.: user program, application program) werden in der Regel in einer höheren Programmiersprache wie C++, Cobol, PL/1, Java usw. geschrieben.

Ein Compiler ist ein Programm, dessen Ausführung ein Quellprogramm (Source Programm) als Input Daten benutzt, um daraus ein Maschinenprogramm, bestehend aus einzelnen Maschinenbefehlen, zu erstellen. Andere Bezeichnungen: Object Programm oder Binaries. Der Aufruf eines Compilers bewirkt eine Übersetzung (Compilation) eines Quellprogramms in ein Maschinenprogramm.

Meistens benutzt man ein weiteres Programm, einen Linker, um das übersetzte Maschinenprogrammen mit anderen, bereits früher übersetzten Programmen, zu einem ausführbaren Programm (executable program) zu verknüpfen. Ein Benutzerprogramm kann ans einzelnen Teilen (Modulen) bestehen, die einzeln geschrieben und übersetzt werden. Die übersetzten Maschinenprogramm Module werden in einem als Programmbibliothek (Library) bezeichneten Verzeichnis auf dem Plattenspeicher untergebracht.

Das Betriebssystem stellt zahlreiche weitere Maschinenprogramm Module zur Verfügung, die mit Hilfe des Linkers ebenfalls mit einem neu erstellten Maschinenprogramm verknüpft werden. Beispiele sind Ein/Ausgabe (I/O) Routinen oder die Socket Library für Intersytem Communication. Unter dem z/OS Betriebssystem stehen viele solcher Maschinenprogramme in einer mit dem Parameter SYS1 gekennzeichneten Library.

Ein Loader ist ein Maschinenprogramm, welches ein ausführbares Programm aus einer Library ausliest und in den Hauptspeicher lädt.

2.1.3 Hauptspeicheradressen

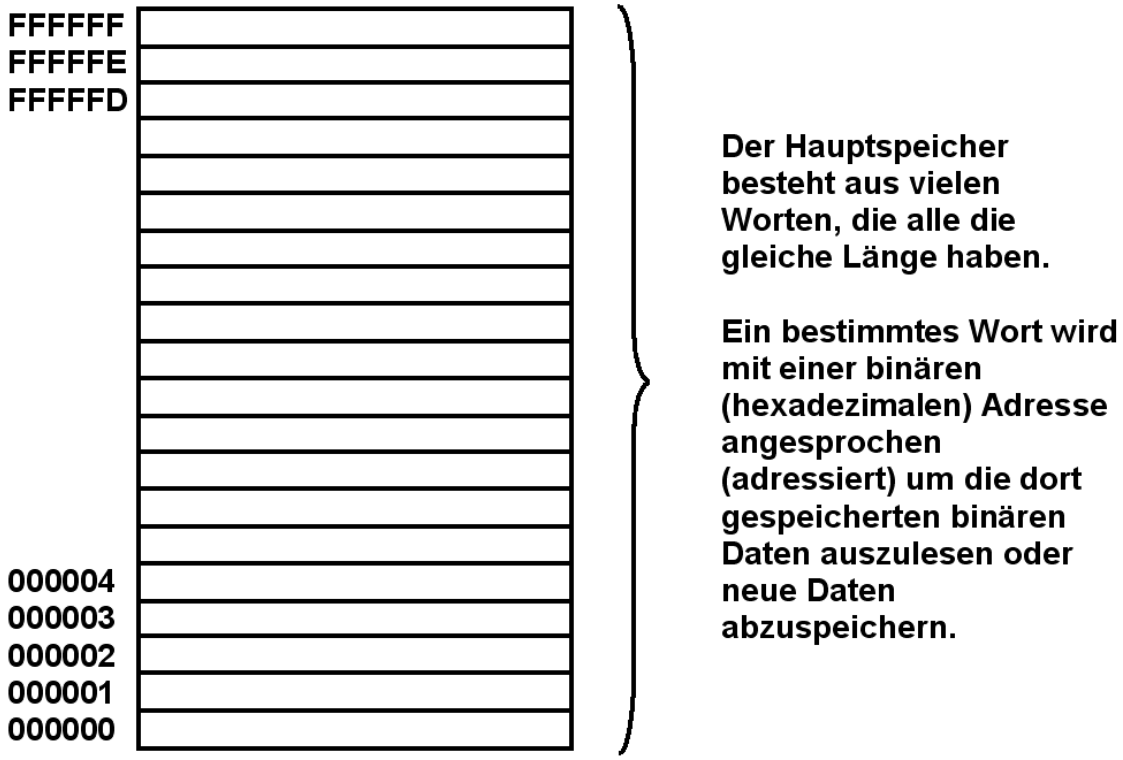

Abb. 2.1.3
Adressierung des Hauptspeichers

Der Hauptspeicher (main Storage) besteht aus lauter gleich langen Wörtern, die mit binären bzw. hexadezimalen Ziffern angesprochen (adressiert), gelesen und gespeichert werden.

Die Wortlänge eines Hauptspeichers ist entweder 8, 12, 16, 14, 32, 36 oder 48 Bit. Moderne Rechner benutzen fast ausschließlich eine Wortlänge von 8 Bit (ein Byte).

Im Hauptspeicher werden sowohl Maschinenprogramme als auch Daten gespeichert.

2.1.4 32 und 64 Bit Adressen

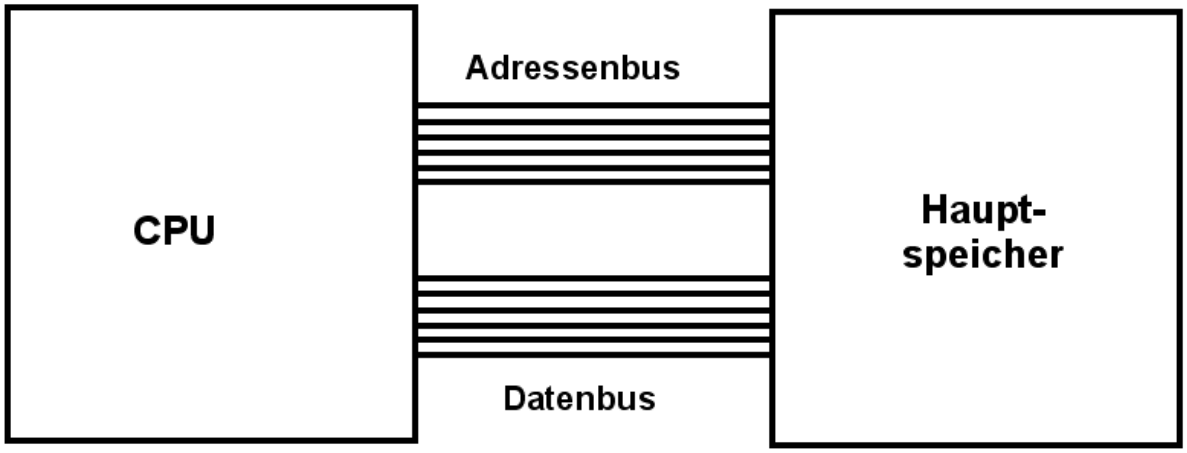

Abb. 2.1.4
Adressierung des Hauptspeichers

Wir unterscheiden zwischen 32 Bit und 64 Bit Architekturen. Bei einer 32 Bit Architektur ist die CPU mit dem Hauptspeicher mit einem Adressenbus und einem Datenbus verbunden. Der Adressenbus besteht aus 32 Leitungen, um eine von 2^{32} Adressen an den Hauptspeicher zu übergeben. Bei einer 64 Bit Architektur sind es 64 Leitungen, um eine von 2^{64} Adressen an den Hauptspeicher zu übergeben. Der Datenbus besteht aus 8 Leitungen, auf denen jeweils 1 Byte (8 Bit) zwischen Hauptspeicher und CPU übertragen wird.

Die Maschinenbefehle gehen von dieser Struktur aus, die wir als das logische Erscheinungsbild einer CPU bezeichnen. Die physische Implementierung eines Rechners weicht in der Regel von der logischen Sicht ab. So besteht z.B. ein Datenbus häufig aus 256 Leitungen, und es werden gleichzeitig 8 Bytes (256) Bits zwischen CPU und Hauptspeicher übertragen. Ein moderner Mainframe Rechner (z196) hat einen physischen Hauptspeicher mit einer maximalen Größe von 4 TByte (2^{42} Byte), für dessen Adressierung 42 Leitungen des Adressenbusses ausreichen.

Anmerkung: Bei der Mainframe Architektur verfügt die ehemalige 32 Bit Architektur nur über 31 Bit Adressen, und adressiert damit einen Hauptspeicher mit einer maximalen Größe von 2^{31} Adressen und 2^{31} Byte Speicherkapazität.

Im Gegensatz zum Hauptspeicher ist der Zugriff auf einen Festplattenspeicher sehr viel komplizierter. Für den Zugriff setzt die CPU ein spezielles Maschinenprogramm, einen I/O (Input/Output) Driver, ein.

Eine moderne CPU führt größenordnungsmäßig eine Milliarde (10^9) Maschinenbefehle pro Sekunde aus. Die Zugriffszeit zum Hauptspeicher (Cache, siehe Abb. 2.4.2) beträgt größenordnungsmäßig wenige 100 Picosekunden (10^{-9} bis 10^{-10} s, logische Sicht). Die Zugriffszeit zum Plattenspeicher beträgt größenordnungsmäßig 10 Millisekunden (10^{-2} s) . Dieser Unterschied in der Zugriffszeit hat einen sehr großen Einfluss auf die Struktur moderner Betriebssysteme.

2.1.5 Supervisor and User Space

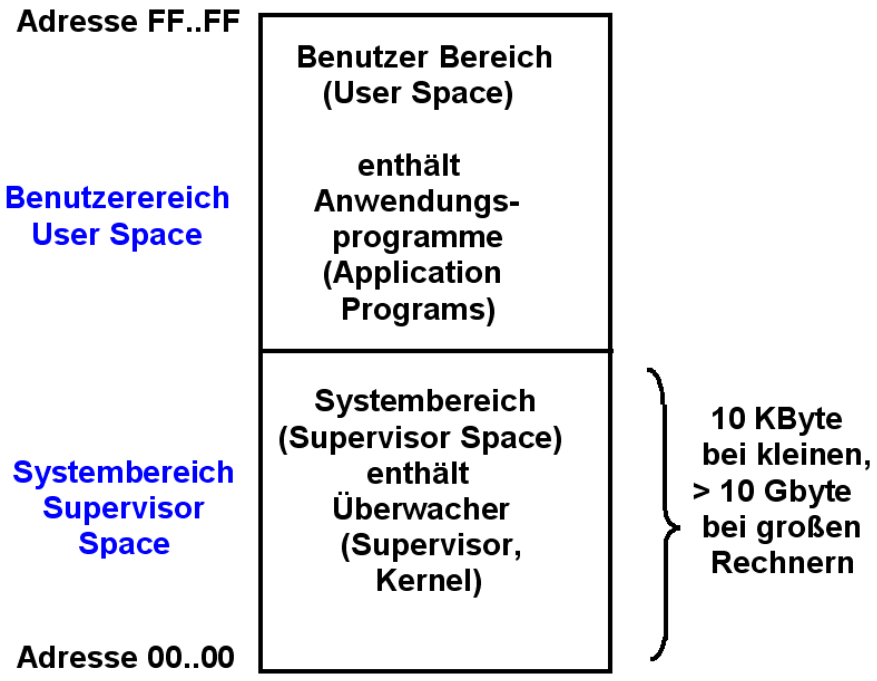

Abb. 2.1.5
Aufteilung des Adressenraums

Ein Betriebssystem enthält Programmroutinen, die von Benutzerprogrammen immer wieder gebraucht werden. Ein Beispiel ist der I/O Driver (Input/Output Driver), der das Lesen oder Schreiben von Daten vom/zum Plattenspeicher bewirkt.

Der Code des Betriebssystems befindet sich auf dem Plattenspeicher. Beim Hochfahren eines Rechners wird ein Teil des Betriebssystems in den Hauptspeicher geladen. Diesen Teil bezeichnet man als Betriebssystem-Kern, Kernel oder Überwacher (Supervisor). z/OS benutzt auch die Begriffe Nucleus oder „Basic Control Program“ (BCP).

Wir bezeichnen die Menge aller Hauptspeicheradressen als Adressenraum (Address Space). In dem hier dargestellten einfachsten Fall wird der Adressenraum des Hauptspeichers in 2 Teile aufgeteilt: einen Teil für den Überwacher (Supervisor Address Space) und einen zweiten Teil für das auf dem Rechner laufende Benutzerprogramm (Anwendungsprogramm) und seine Daten (User Address Space). Das ursprüngliche DOS Betriebssystem für den PC hatte diese Struktur.

Damit ein fehlerhaftes oder böswilliges Benutzerprogramm nicht unbefugt Programme oder Daten im Überwacher Adressenraum modifizieren kann, läuft der Rechner in jedem Augenblick entweder im Überwacherstatus (Supervisor State) oder im Benutzerstatus (User State, auch als Problem State bezeichnet).

Ein Programm, das im Überwacherstatus läuft, kann auf den gesamten Adressenraum des Hauptspeichers zugreifen. Ein Programm, das im User Modus läuft, hat Zugriffsrechte nur für den User Adressenraum.

2.1.6 Prozess

Was ist ein Prozess ?

Ein Maschinen-Programm lässt sich mit den Noten eines Musikstückes vergleichen. Ein Konzert ist die Aufführung eines Musikstückes entsprechend den verwendeten Musiknoten. Ein Prozess ist die Ausführung eines Maschinenprogramms auf einem Computer.

Der Begriff „Prozess“ ist ein zentrales Konzept in der modernen Datenverarbeitung. Ein Prozess ist die Ausführung von einem Programm (oder mehreren Programmen) um ein für einen Benutzer relevantes Problem zu bearbeiten. Beispiele für Prozesse sind die monatliche Lohn- und Gehaltsabrechnung in einem Unternehmen, die Verbuchung bei der Überweisung eines Geldbetrages, eine URL Abfrage im Internet oder die Stücklistenerstellung beim Bau eines Automobils.

Ein Benutzer Prozess läuft innerhalb des Benutzer Adressenraums. Systemprozesse bearbeiten irgendwelche Teilaufgaben des Betriebssystems.

Die Ausführung eines Prozesses bewirkt in der Regel, dass Daten abgeändert werden, die auf einem Festplattenspeicher permanent gespeichert sind. Ein Beispiel sind Kontodaten bei der Überweisung eines Geldbetrages. Hierzu lädt der Prozess die Daten vom Plattenspeicher in den Hauptspeicher, modifiziert sie und schreibt das Ergebnis auf den Plattenspeicher zurück.

Eine Transaktion ist eine spezielle Art eines Prozesses, bei dem garantiert wird, das alle Änderungen von Daten auf dem Plattenspeicher entweder vollständig oder gar nicht, nicht aber teilweise, erfolgen.

Ein Prozess wird von z/OS als „Task“ bezeichnet. Eine Task wird durch einen Bereich im Hauptspeicher definiert, der als „Task Control Block (TCB) bezeichnet wird. Linux verwendet hier die Bezeichnung „Process Control Bloch (PCB).

2.1.7 Struktur eines Benutzerprogramms

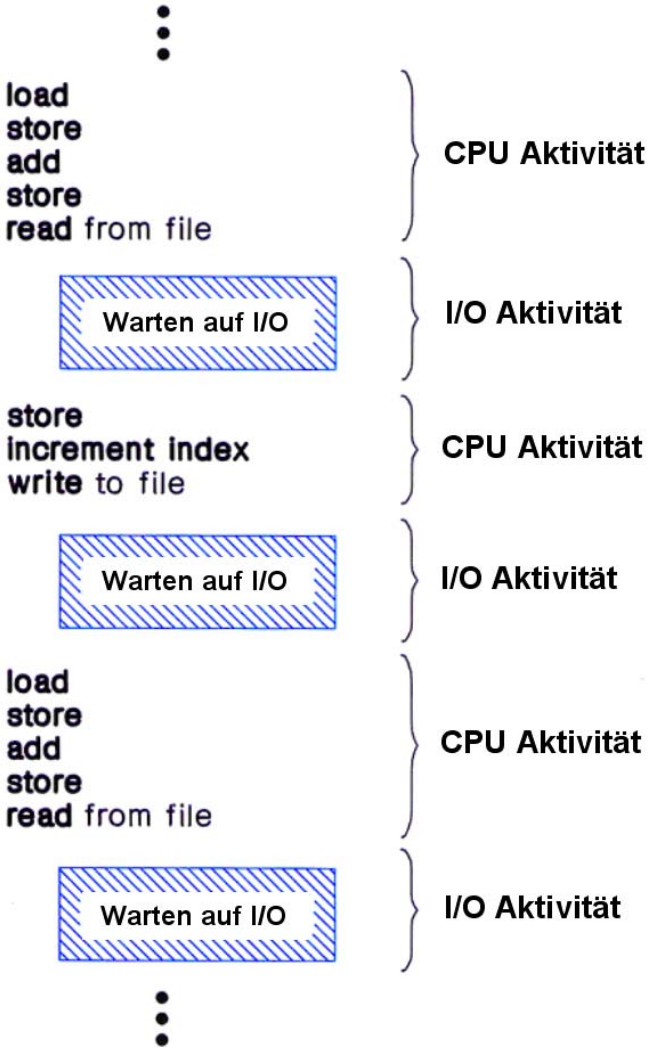

Abb. 2.1.6
Struktur eines Benutzerprogramms

Ein Benutzerprogramm besteht aus einer Folge von

- **Gruppen von Maschinenbefehlen**
- **I/O Operationen (Zugriff auf Festplattenspeicher-Daten).**

Die Programmausführung ist eine abwechselnde Folge von CPU und I/O Aktivitäten.

Beim Start einer I/O Operation muss das Anwendungsprogramm in der Regel warten, bis die I/O Operation abgeschlossen ist (z.B. die vom Plattenspeicher gelesenen Daten im Hauptspeicher eingetroffen sind).

Der Zugriff auf einem Plattenspeicher benötigt etwa 10 Millisekunden. Während dieser Zeit könnte die CPU etwa 10 Millionen Maschinenbefehle ausführen. Es wäre unökonomisch, wenn die CPU während der I/O Aktivität warten müsste.

2.1.8 Multiprogrammierung

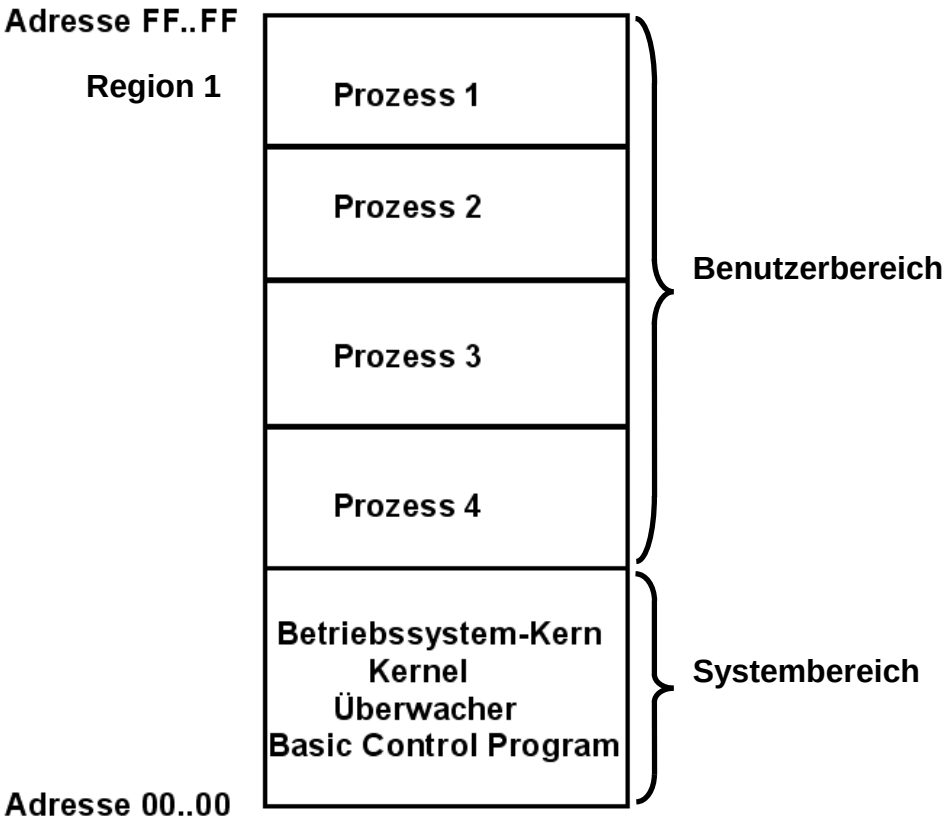

Abb. 2.1.7
Multiprogrammierung

Bei der Multiprogrammierung teilt man des Benutzer Adressenraum in mehrere Teilbereiche (Regions) auf und ordnet einem Prozess jeweils eine eigene Region zu.

Jetzt befinden sich die Programme (Code) mehrerer Prozesse gleichzeitig im Hauptspeicher. Wir nehmen in diesem Beispiel an, dass der Rechner eine einzige CPU hat. Wenn nun ein Prozess eine I/O Operation ausführt, wird er in einen Wartezustand versetzt. Ein anderer Prozess, dessen Programme und Daten sich ausführungsbereit in seiner Region des Benutzer Adressenraums befinden, wird nun von der CPU ausgeführt, bis er ebenfalls eine I/O Operation ausführen will. Jetzt beginnt die CPU einen dritten Prozess auszuführen usw.

In einem Mainframe Rechner werden auf diese Art typischerweise Tausende oder Zehntausende von Prozessen parallel ausgeführt. Diese Art der Datenverarbeitung, bei der mehrere Prozesse ineinander geschachtelt parallel verarbeitet werden, wird als „Multiprogrammierung“ (Multiprogramming) bezeichnet.

Prozesse werden normalerweise gestartet, verrichten ihre Arbeit und werden wieder beendet. Ein Daemon ist ein lang laufender Prozess, der häufig zusammen mit dem Hochfahren des Betriebssystems gestartet, und nie beendet wird.

2.1.9 Zustand von Prozessen

Wenn ein Prozess eine Plattenspeicher I/O Operation startet wird er in den Zustand „wartend“ (waiting) versetzt. Wenn die I/O Operation abgeschlossen ist, könnte er wieder von der CPU ausgeführt werden. Da die CPU vermutlich gerade mit einem anderen Prozess beschäftigt ist, wird er statt dessen in den Zustand „ausführbar“ (ready) versetzt. Irgendwann wird die CPU den Prozess weiter verarbeiten. Ein Prozess, der durch die CPU verarbeitet wird, wird als „laufend“ (running) bezeichnet.

Jeder Prozess rotiert wiederholt durch die Zustände laufend, wartend und ausführbar. Moderne Rechner haben meistens mehr als eine CPU, z.B. 101 bei einem zEC12 Mainframe. In der Regel verarbeitet jede CPU einen anderen Prozess. In diesem Fall können sich mehrere Prozesse im Zustand „laufend“ befinden.

- **Ein Prozess befindet sich im Zustand laufend, wenn er von einer der verfügbaren CPUs ausgeführt wird.**
- **Ein Prozess befindet sich im Zustand wartend, wenn er auf den Abschluss einer I/O Operation wartet.**
- **Ein Prozess befindet sich im Zustand ausführbar, wenn er darauf wartet, dass die Scheduler Komponente des Überwachers ihn auf einer frei geworden CPU in den Zustand laufend versetzt.**

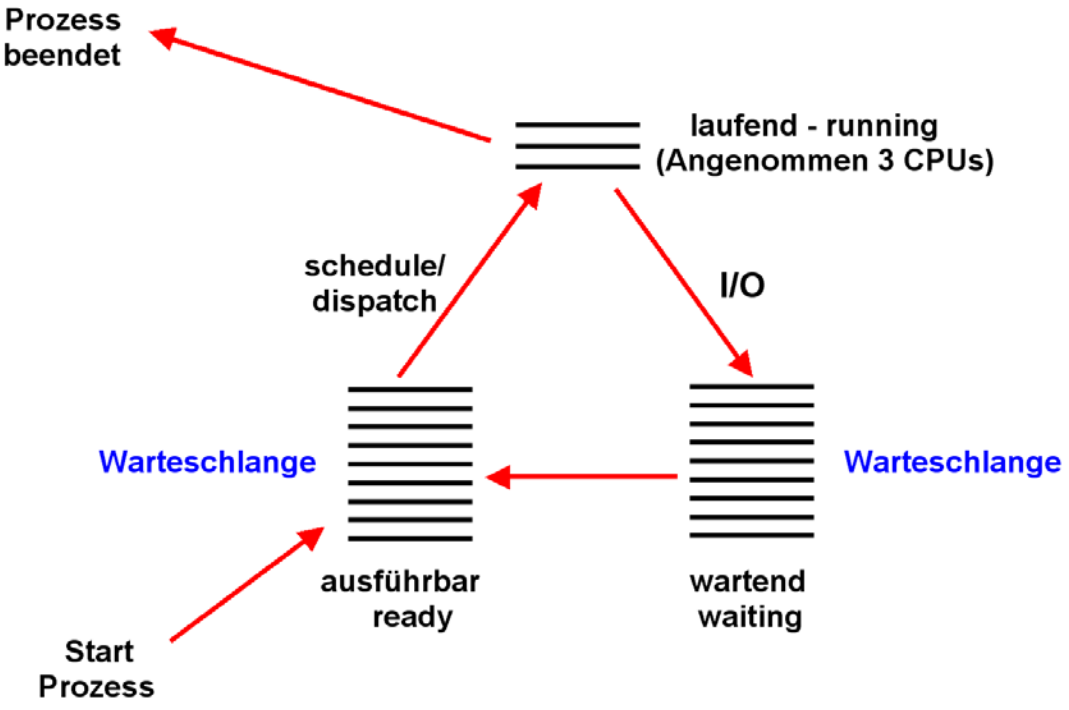

Abb. 2.1.8
Zustand von Prozessen

Jede Linie entspricht einem Prozess. Jeder Prozess wird durch einen Bereich im Hauptspeicher (Task Control Block, TCB) gekennzeichnet und repräsentiert. Die Scheduler Komponente des Überwachers unterhält drei Warteschlangen (Queues) für wartende, ausführbare und laufende Prozesse. In diese Warteschlangen werden die TCBs der Prozesse beim Wechsel des Zustandes eingeordnet.

Die Scheduler Komponente des Überwachers selektiert aus der Warteschlange ausführbarer Prozesse jeweils einen Kandidaten wenn immer eine CPU frei wird.

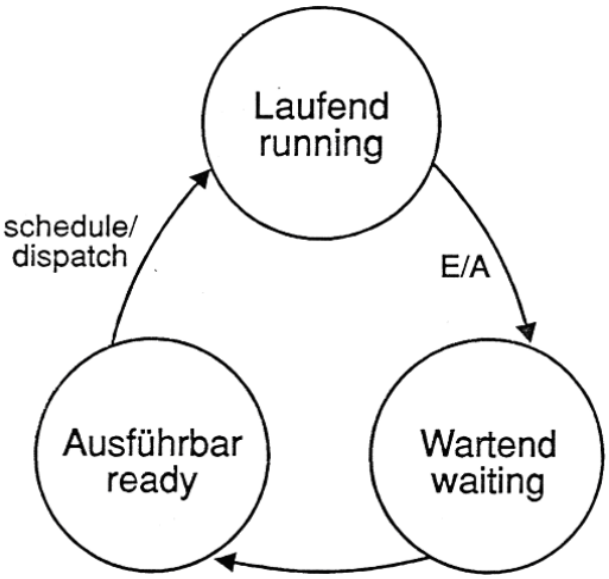

Abb. 2.1.9
Prozess Zustandsdiagramm

Dies ist eine andere Darstellung der obigen Abb. 2.1.8 Auf einem Mehrrechnersystem (Rechner mit mehr als einer CPU) können sich mehrere Prozesse im Zustand laufend befinden. Es ist auch möglich, dass ein Prozess mehrere CPUs beschäftigt.

Ausführbare Prozesse verfügen über Ressourcen, besonders über Platz im Hauptspeicher (Programm Code, Daten)

2.1.10 Zeitscheiben Steuerung (Time Slicing)

Bei der Ausführung eines Anwendungsprogramms treten I/O Anforderungen relativ häufig auf. In diesem Fall wird der betroffene Prozess in den Zustand wartend versetzt, und die Scheduler Komponente des Überwachers (Supervisor) selektiert aus der Menge (Queue) der ausführbaren Prozesse einen Prozess, der in den Zustand laufend versetzt wird.

In der großen Mehrzahl der Fälle wird dieser Prozess nach einem Zeitraum, der selten 1 ms überschreitet, in den Zustand wartend versetzt, weil er eine I/O Operation ausführt. Es gibt jedoch Situationen, in denen dies nicht der Fall ist. Um zu verhindern, dass ein Prozess eine CPU usurpiert, enthält der Scheduler eine Zeitscheiben (Time Slice) Steuerung, die einen laufenden Prozess nach einer gewissen Zeit (typischerweise wenige ms) unterbricht, in den Zustand ausführbar versetzt, und dafür einen anderen ausführbaren Prozess in den Zustand laufend versetzt. Dieser Vorgang wird als „Preemption“ bezeichnet.

2.1.11 Multiprogrammierung

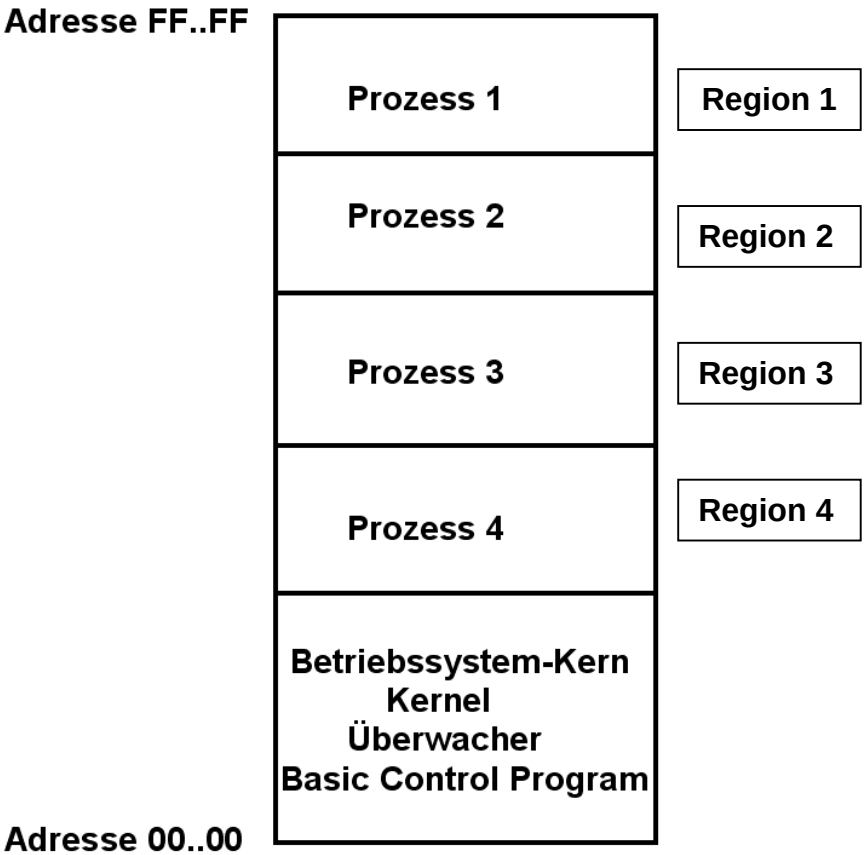

Abb. 2.1.10
Multiprogrammierung

Die Aufteilung des Benutzer Adressenraum in mehrere Regions für mehrere Prozesse erzeugt mehrere Probleme:

- **Es muss verhindert werden, dass fehlerhafte Programme eines Prozesses auf die Region eines anderen Prozesses zugreift.**

- **Bei Beendigung eines Prozesses wird die benutzte Region des Benutzer Adressenraums freigegeben. In ihr kann jetzt ein neuer Prozess gestartet werden.**

  Die einzelnen Prozesse benötigen aber unterschiedlich viel Speicherplatz. Die Regions haben deshalb eine unterschiedliche Größe. Die Größe der freigewordenen Region entspricht nicht notwendigerweise den Bedürfnissen des neuen Prozesses. Eine komplexe Speicherverwaltungskomponente (als Teil des Überwachers) ist erforderlich.

Zur Lösung dieses Problems führte IBM 1972 die virtuelle Speichertechnik (Virtual Storage) für eine neue Serie von Mainframe Rechnern (System /370) ein.

2.2 Virtual Storage

2.2.1 Virtueller Speicher

Die Befehlsadressen und effektiven Adressen der Operanden arbeiten bei einem modernen Rechner mit einer Illusion eines Speichers (virtueller Speicher), der eine andere und einfachere Struktur hat als der reale Hauptspeicher, in dem sich die Befehle und Operanden tatsächlich befinden.

Hierzu wird der virtuelle Speicher in Blöcke aufgeteilt, die alle die gleiche Größe (z.B. 4096 Bytes) haben. Diese Blöcke nennt man Seiten (pages).

Der Hauptspeicher wird in Blöcke aufgeteilt, die genauso groß wie die Seiten sind und als Platzhalter für die Aufnahme von Seiten dienen. Diese Blöcke nennt man Rahmen (frames oder pageframes).

Virtuelle Adressen adressieren Maschinenbefehle und Operanden im virtuellen Speicher.

Reale Adressen adressieren Maschinenbefehle und Operanden im (realen) Hauptspeicher.

Der virtuelle Speicher ist aus der Sicht des Programmierers ein kontinuierlicher, einfach zusammen-hängender, linearer Adressenraum.

Die Abbildungsvorschrift für die virtuelle (logische) in die reale (Physikalische) Adressumsetzung ist in einer Seitentabelle enthalten.

Wir benutzen die folgenden Begriffe:

Ein Adressenraum (Address Space) ist die lineare Folge von Adressen der Bytes (oder anderer adressierbarer Einheiten) eines Speichers.

Ein Virtueller Adressenraum ist die lineare Folge von Adressen der Bytes eines virtuellen Speichers.

Ein Realer Adressenraum ist die lineare Folge von Adressen der Bytes eines realen (tatsächlich existierenden) Hauptspeichers.

Mittels der Adressumsetzung (Dynamic Address Translation, DAT) wird der virtuelle Adressenraum der Befehle und Operanden eines Benutzerprozesses vom realen Adressenraum des Hauptspeichers getrennt. Die Adressumsetzung bewirkt die Abbildung von virtuellen Adressen auf reale Adressen.

2.2.2 Seiten und Rahmen

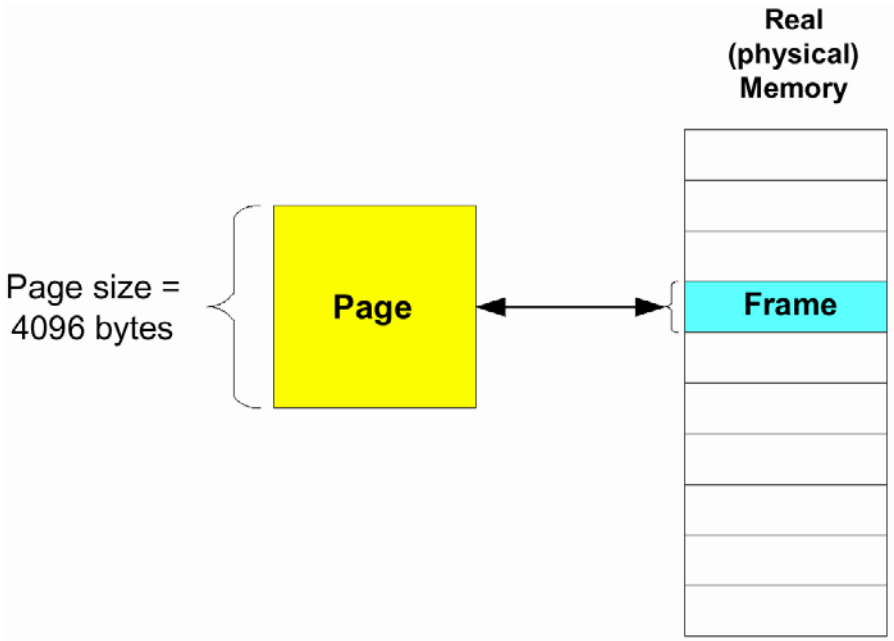

Abb. 2.2.1
Seiten und Rahmen

Für die Adressumsetzung wird der Virtuelle und der reale Adressenraum jeweils in 4096 Byte große Blöcke aufgeteilt. Die Blöcke des virtuellen Speichers werden als Seiten (Pages) und die Blöcke des realen Speichers als Rahmen (Frames) bezeichnet. Die Anordnung der Bytes innerhalb einer Seite oder eines Rahmens ist identisch und wird bei der Adressumsetzung nicht verändert. Es erfolgt aber eine willkürliche Zuordnung von Seiten- zu Rahmenadressen.

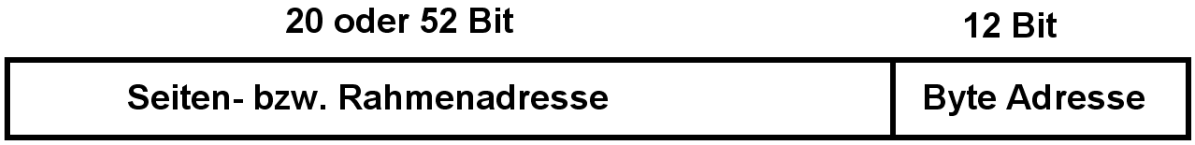

Abb. 2.2.2
Adressenformat

Eine virtuelle bzw. reale Adresse besteht deshalb grundsätzlich aus zwei Feldern:

- **Seiten- bzw. Rahmenadresse und**
- **Byteadresse**

2.2.3 Seitentabelle

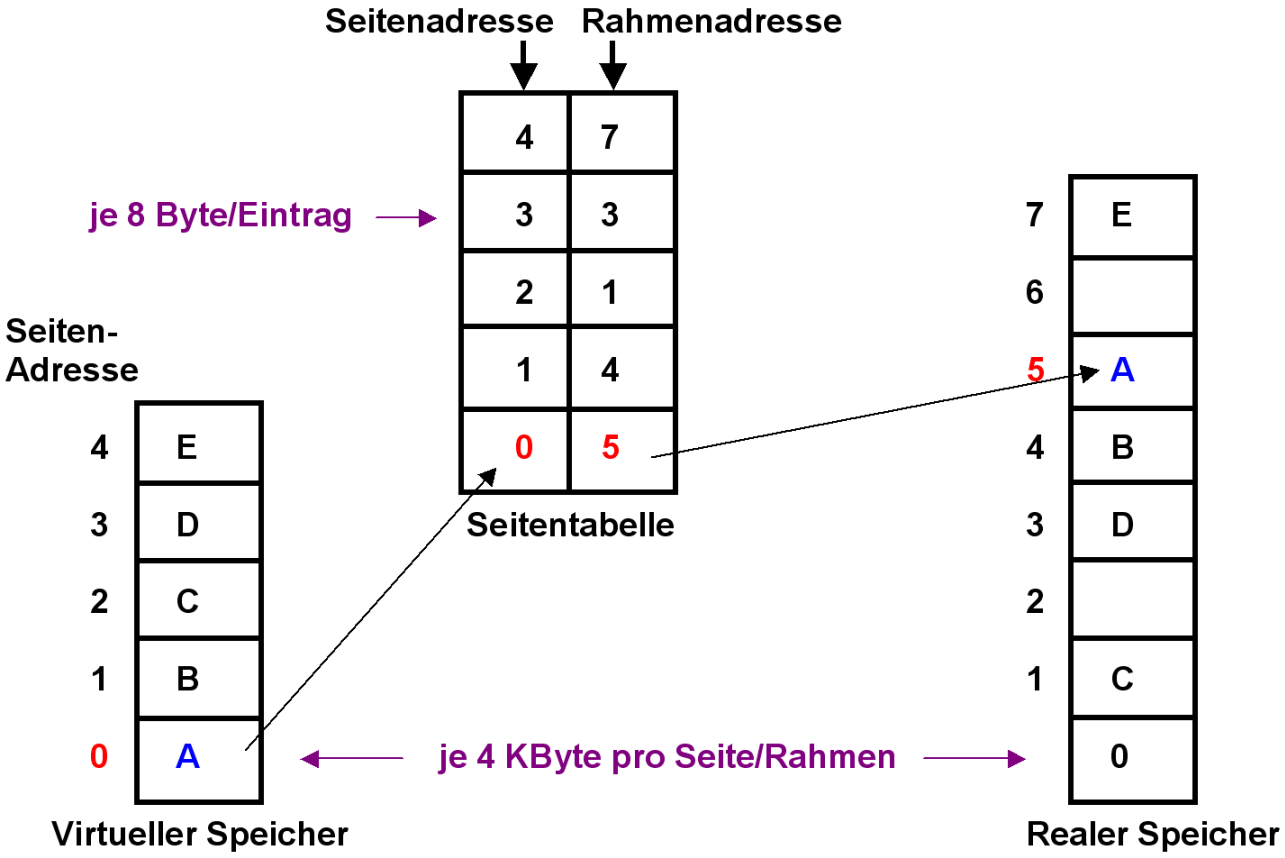

Abb. 2.2.3
Aufgabe der Seitentabelle

In dem hier gezeigten Beispiel erstreckt sich der virtuelle Speicher über 5 Seiten, der reale Hauptspeicher über 8 Rahmen. Virtuelle und reale Speicher können durchaus unterschiedliche Größen haben.

Die 5 Seiten des virtuellen Speichers speichern die Inhalte A, B, C, D und E. A hat die Seitenadresse 0. Die Adressumsetzung erfolgt mit Hilfe einer Seitentabelle (page table). Jede Seite des virtuellen Speichers hat einen Eintrag in der Seitentabelle. Der Eintrag für Seitenadresse 0 besagt, dass der Seiteninhalt A in dem Rahmen mit der Rahmen-Adresse 5 abgebildet wird.

2.2.4 Größe des virtuellen Speichers

Theoretisch ist es denkbar, dass bei einem Rechner mit 64 Bit Adressen jeder virtuelle Adressenraum eine Größe von 2^{64} Bit hat. In der Praxis ist das nicht machbar, unter anderem, weil die Größe der Seitentafel im realen Speicher proportional zu der Größe aller virtuellen Speicher ist.

Seitentabellen erfordern wenige Promille der virtuellen Speichergröße an realem Hauptspeicherplatz. Zwei Promille von 2^{64} Bytes = 16 Exabyte sind 32 Terabyte.

Ein Mainframe Rechner kann 10 000 Prozesse mit getrennten Seitentabellen unterhalten. Die maximale Hauptspeichergröße eines z196 Mainframe beträgt aber „nur“ 3 TByte.

Virtuelle Speicher sollen deshalb nur so groß eingerichtet werden, wie es der Prozess erfordert.

Die Seitentabelle befindet sich (entweder teilweise oder ganz) im Überwacherteil des realen Hauptspeichers.

Genau genommen verwendet ein Mainframe (und auch ein x86) Rechner nicht eine einzige Seitentabelle, sondern eine 2 – 5 stufige Hierarchie von Seitentabellen. Im Gegensatz dazu verwendet die PowerPC Architektur nur eine einzige Seitentabelle

Abb. 2.2.4 zeigt die 2-stufige Adressumsetzung, wie sie in den 32 (31) Bit Versionen der x86 und der System z Architektur implementiert ist.

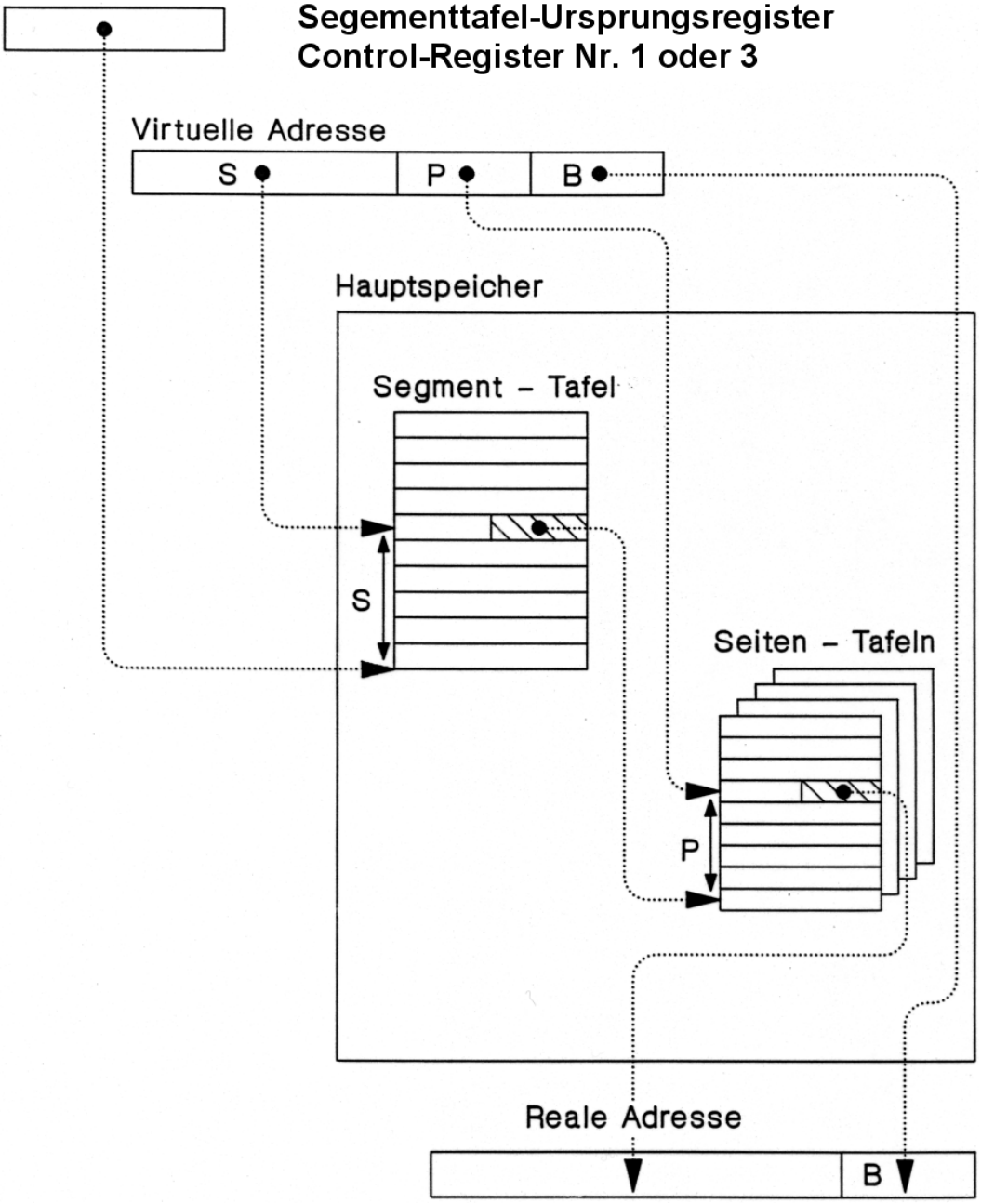

Abb. 2.2.4
Adressumsetzung Pentium, S/390

Die Adressumsetzung erfolgt durch zwei Arten von Tabellen (Segment- und Seitentabelle), die im Kernel Bereich des Hauptspeichers untergebracht sind (Siehe auch Band 1, Abschnitt 2.2.4). Die Anfangsadresse der Segmenttabelle steht in einem Control Register der Zentraleinheit, z.B. CR Nr. 1 bei der S/390 Architektur.

Die Adressumsetzung erfolgt bei jedem Hauptspeicherzugriff durch Hardware mit Unterstützung durch die Segmenttabelle und eine Seitentabelle im Kernel-Bereich. Sie kann durch den Programmierer nicht beeinflusst werden.

(Zur Leistungsverbesserung werden die derzeitig benutzten Adressen in einem Adressumsetzpuffer untergebracht.)

2.2.5 Externer Seitenspeicher

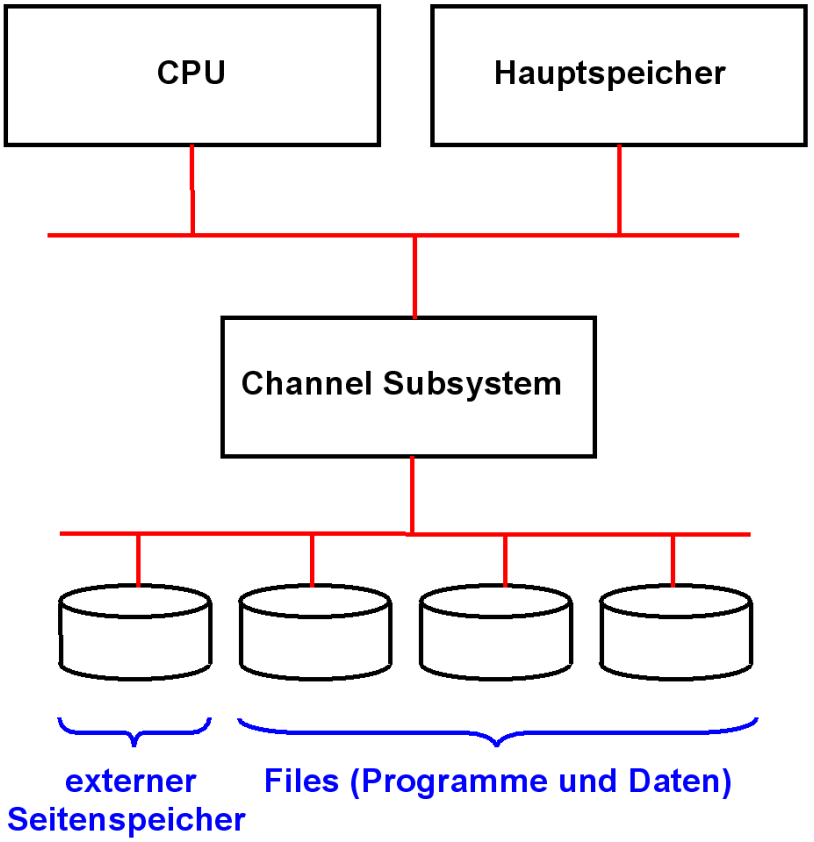

Abb. 2.2.5
Externer Seitenspeicher

Der virtuelle Speicher kann auch größer als der reale Hauptspeicher sein.

In diesem Fall wird ein Teil des realen Hauptspeichers (Real Storage) auf einen als externer Seitenspeicher (Auxiliary Storage oder Page Datasets) bezeichneten Plattenspeicher ausgelagert.

Bei einem Windows System ist dies z.B. der Bereich pagefile.sys auf der Partition C: . Bei einem Mainframe ist das in der Regel ein (oder mehrere) nur hierfür benutzter Festplattenspeicher. Der externe Seitenspeicher ist hierfür in 4096 Byte große Rahmen aufgeteilt.

Sehr grob betrachtet hat das Channel Subsystem eines Mainframes die Funktion eines I/O Adapters.

2.2.6 Speichern der Rahmen

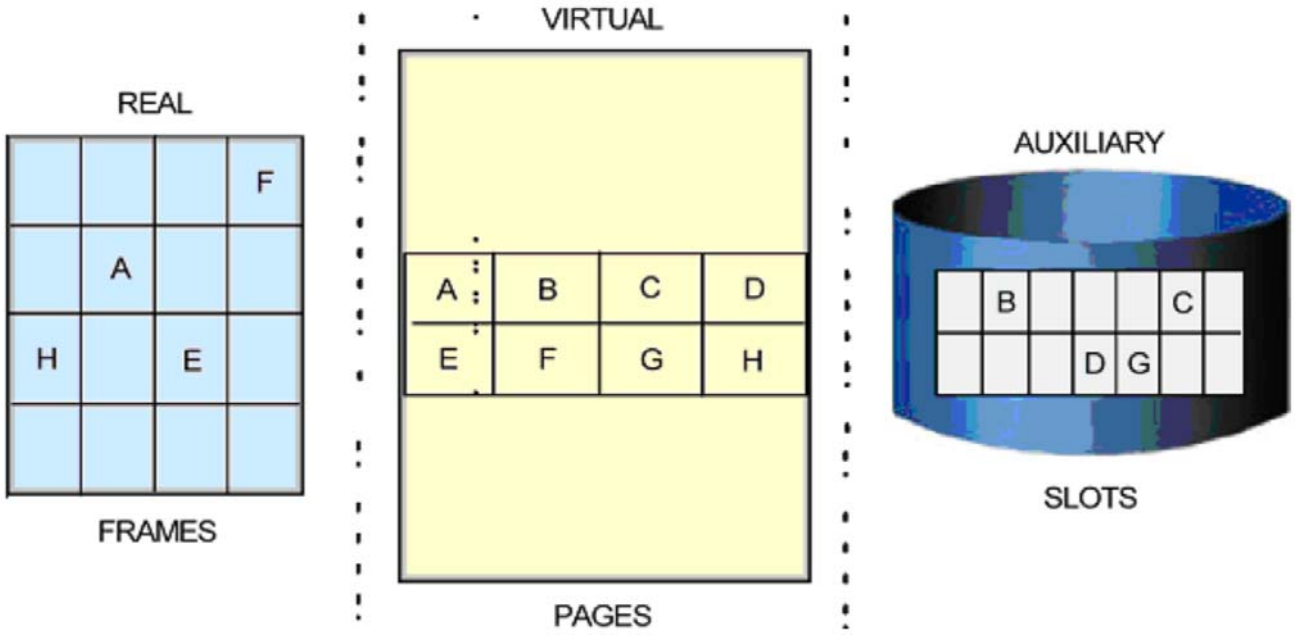

Abb. 2.2.6
Speichern der Rahmen

In dem hier gezeigten Beispiel enthält der virtuelle Speicher die Seiten mit den Inhalten A, B, C, D, E, F, G und H.

A, E, F und H sind in Rahmen des realen Hauptspeichers (Real Storage) abgespeichert. B, C, D und G befinden sich auf dem externen Seitenspeicher (Auxiliary Storage). z/OS bezeichnet die Rahmen des externen Seitenspeichers gelegentlich auch als „Slots“.

Wenn ein Benutzerprogramm auf B, C, D oder G zugreifen will, muss eine Komponente des Überwachers, der „Seitenüberwacher“ (Paging Supervisor), den Rahmen zuerst vom externen Seitenspeicher in den Hauptspeicher kopieren. Vermutlich muss er, um Platz zu schaffen, vorher den Inhalt einen anderen Rahmen des Hauptspeichers auf den externen Seitenspeicher auslagern.

2.2.7 Demand Paging

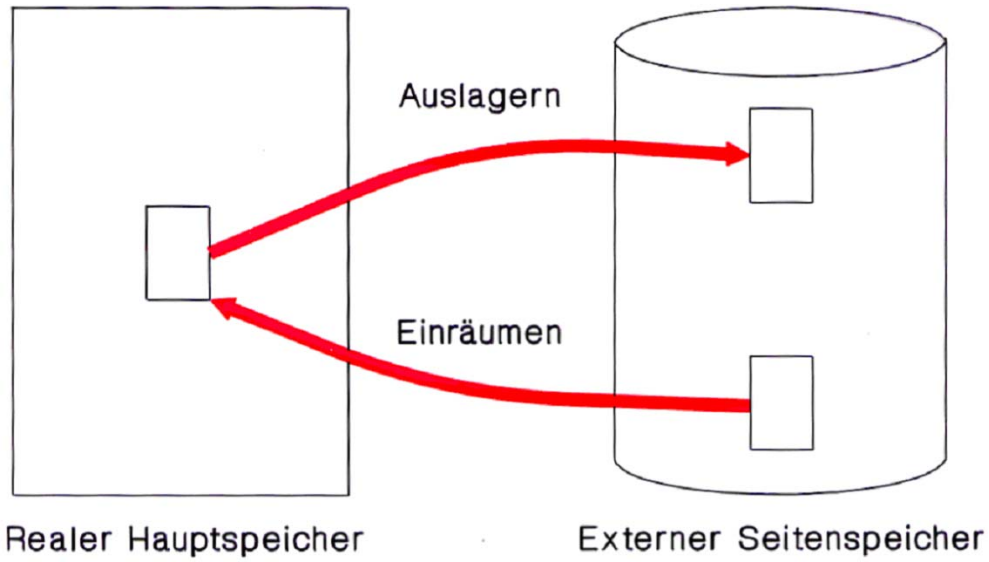

Abb. 2.2.7
Demand Paging

Mehrere Prozesse haben eigene (i n der Regel) unabhängige virtuelle Speicher. Jeder virtuelle Speicher kann größer als der reale Hauptspeicher sein.

Der größere Teil der Seiten eines virtuellen Speichers ist in jedem Augenblick auf einem " externen Seitenspeicher " (Auxiliary Storage), typischerweise ein Festplattenspeicher, ausgelagert. Der reale Speicher besteht somit aus 2 Teilen: dem realen Hauptspeicher und dem externen Seitenspeicher.

Beim Zugriff zu einer ausgelagerten Seite (nicht in einem Rahmen des Hauptspeichers abgebildet) erfolgt eine " Fehlseitenunterbrechung " (Page Fault).

Diese bewirkt den Aufruf einer Komponente des Überwachers (Seitenüberwacher, Paging Supervisor), der die benötigte Seite aus dem externen Seitenspeicher holt und in den Hauptspeicher einliest. Evtl. muss dafür Platz geschaffen werden, indem eine andere Seite dafür auf den externen Seitenspeicher gelegt wird.

Dieser Vorgang wird als „ Demand Paging " bezeichnet.

2.2.8 Abbildung des Virtuellen Speichers auf den realen Speicher

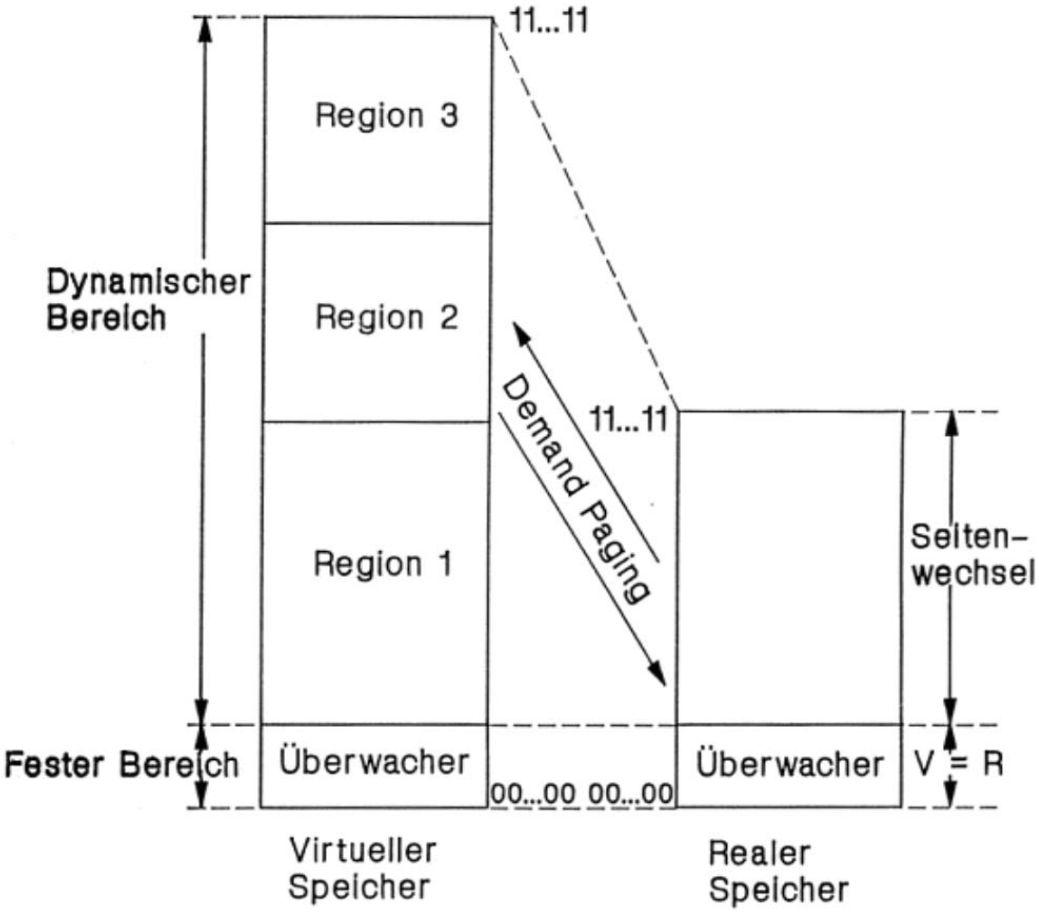

Abb. 2.2.8
Abbildung des virtuellen Speichers

Der Benutzer Adressenraum ist in der Regel größer als der reale Hauptspeicher. Ein Teil des Benutzerraums wirt deshalb in jedem Augenblick auf den externen Seitenspeicher ausgelagert.

Eine Möglichkeit der Nutzung des virtuellen Speicherkonzeptes besteht darin, den virtuellen Speicher in mehrere Regionen aufzuteilen, wobei in jeder Region ein Prozess läuft. Hierbei ist in jedem Augenblick ein Teil der Seiten einer jeden Region im realen Hauptspeicher abgebildet; der Rest befindet sich auf dem externen Seitenspeicher. Der Inhalt der Regionen verändert sich auf Grund des Demand Paging Konzeptes ständig; der Benutzer Adressenraum wird deshalb als der Dynamische Bereich bezeichnet.

Der Inhalt der Rahmen des realen Hauptspeichers ändert sich ständig, da mittels des Demand Paging ständig Seiten zwischen dem Hauptspeicher und dem externen Seitenspeicher hin- und herbewegt werden.

Hierzu gibt es eine Ausnahme: Seiten, die den Überwacher enthalten, werden aus Performance Gründen ständig im Hauptspeicher gehalten. Weiterhin sind die virtuellen Adressen des Überwachers identisch mit den realen Adressen. Man bezeichnet den Überwacher deshalb auch als den Residenten Überwacher. Der Überwacher belegt die untersten Adressen, beginnend mit der hexadezimalen Adresse 000---000 .

Streng genommen ist diese Darstellung stark vereinfacht. Tiefer gehende Details werden aber für die folgenden Erläuterungen nicht benötigt.

2.2.9 Abbildung der virtuellen Adressen

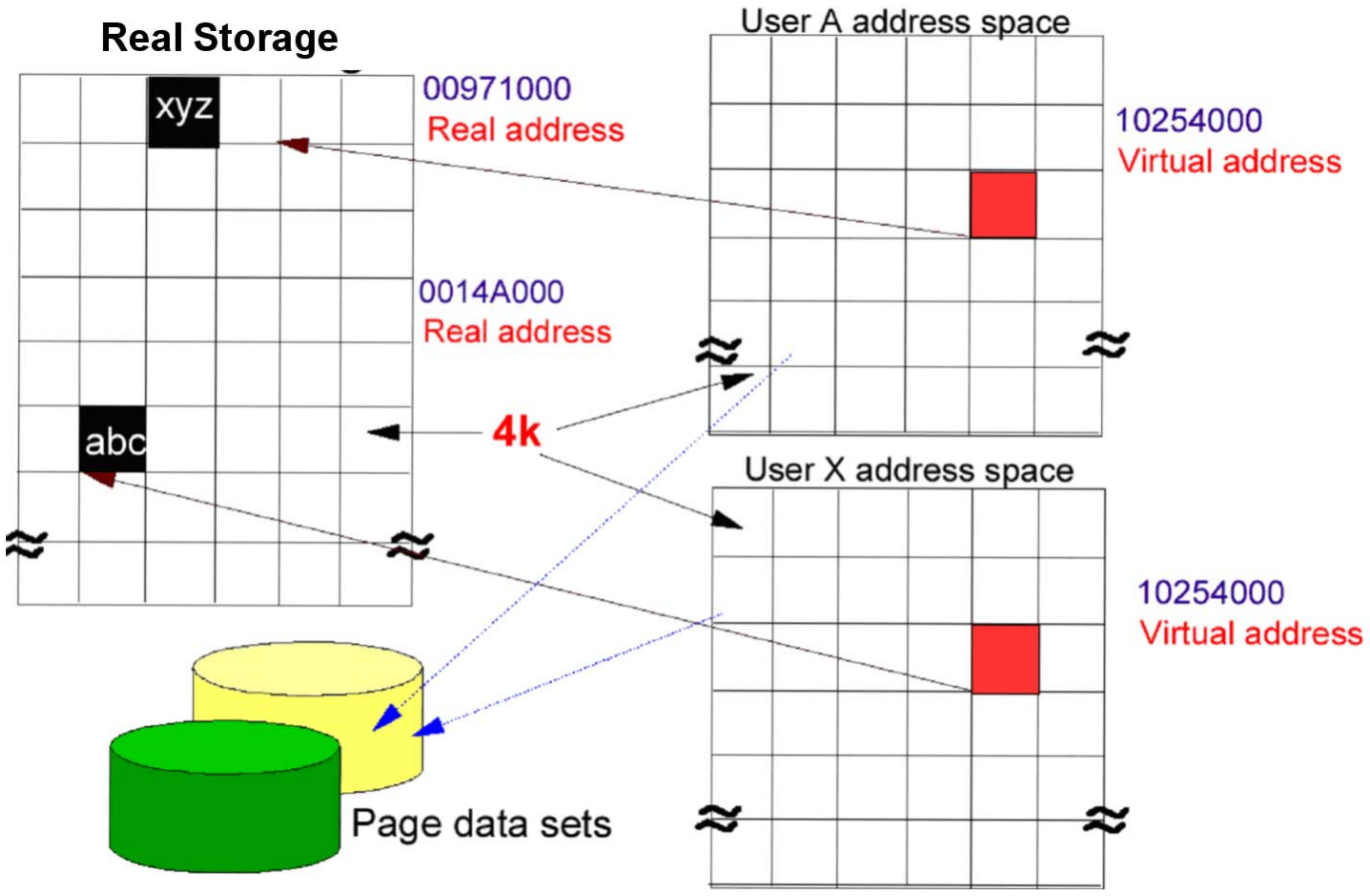

Abb. 2.2.9
Adressenabbildung

Gezeigt ist, wie zwei Seiten in getrennten virtuellen Adressräumen, aber mit identischen virtuellen Adressen, auf unterschiedliche reale Rahmenadressen abgebildet werden.

Seiten werden während der Prozessausführung mehrfach auf den Externen Seitenspeicher ausgelagert und später wieder in den Hauptspeicher eingeräumt. Hierfür werden vermutlich andere Rahmen benutzt; die reale Hauptspeicheradresse ändert sich zwischen Auslagern und Einräumen.

2.2.10 Mehrfache virtuelle Adressenräume

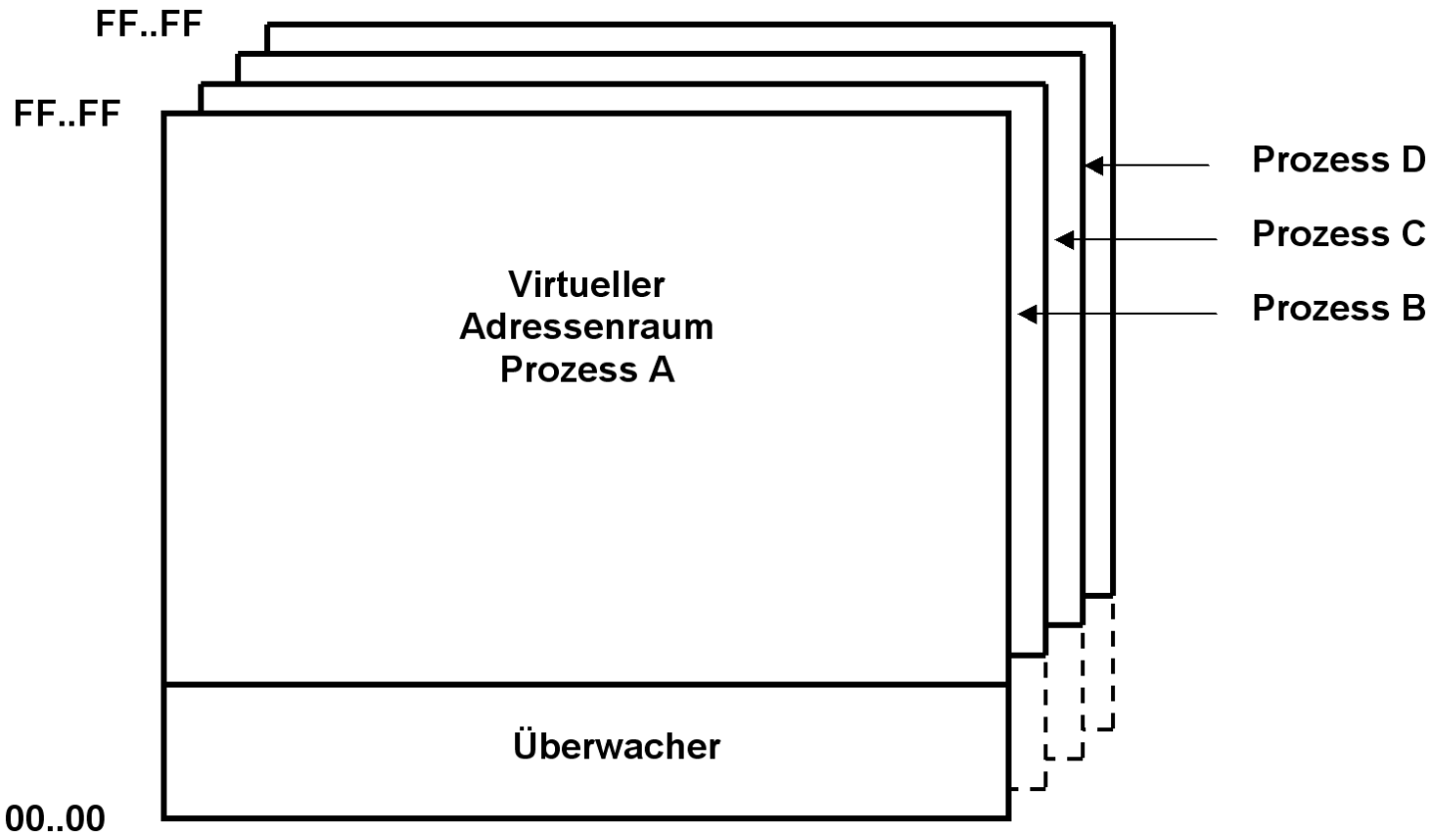

Abb. 2.2.10
Mehrfache virtuelle Adressenräume

Da der virtuelle Speicher sehr viel größer als der reale Hauptspeicher sein kann, ist es möglich, den Prozessen A, B, C und D jeweils einen virtuellen Speicher maximaler Größe zuzuorden. In diesem Fall benutzen unterschiedliche virtuelle Speicher identische Adressen. Der Adressenbereich der virtuellen Speicher geht von Hex 000..000 bis zu einer maximalen Größe, theoretisch bis zu FFF...FFF, z.B. 2^{31} oder 2^{64} Bytes. Da der reale Hauptspeicher viel kleiner ist, wird ein Großteil der Seiten auf dem externen Seitenspeicher abgespeichert.

Jeder Prozess hat einen eigenen virtuellen Adressenraum. z/OS bezeichnet den virtuellen Adressenraum (virtual Address Space) eines Prozesses auch als Region.

2.2.11 Feste Adressen für den Überwacher

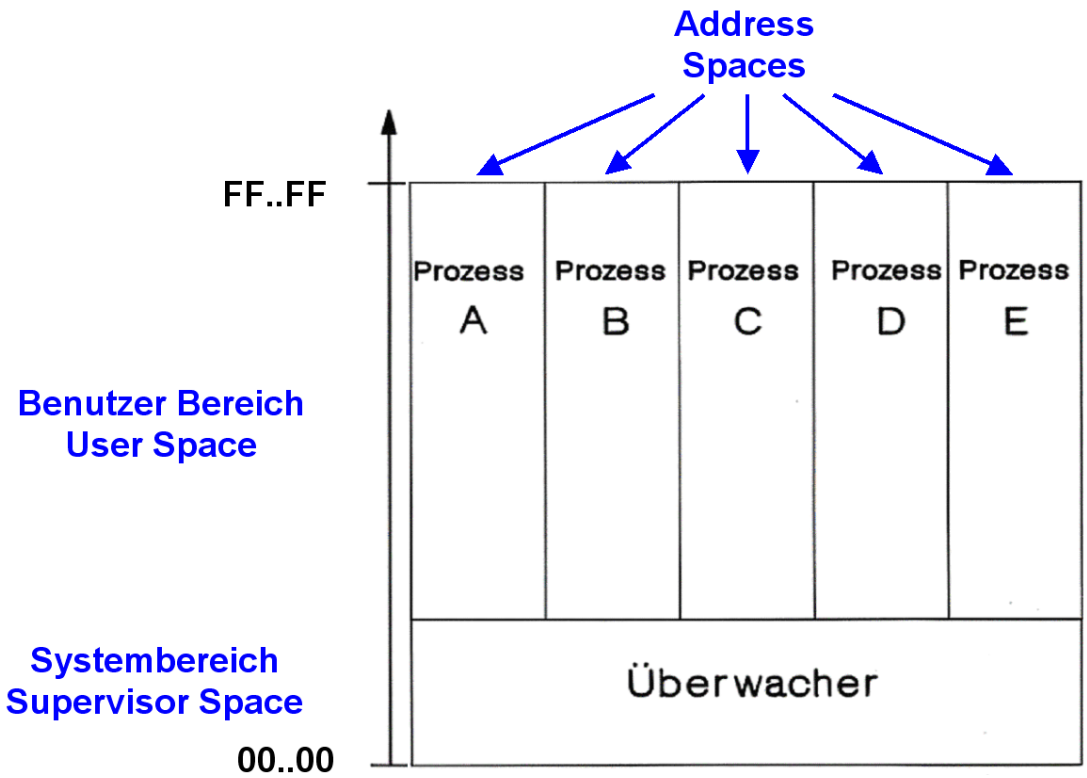

Abb. 2.2.11
Addressspaces können den ganzen Adressenbereich abdecken

Die Abbildung der virtuellen Adressen auf unterschiedliche reale Adressen erfolgt, in dem man jedem Prozess eine eigene Seitentabelle zuordnet.

Der Systembereich (Supervisor Space) mit dem Überwacher ist nur einmal vorhanden, und ist Bestandteil aller virtuellen Adressenräume. Für den Systembereich sind virtuelle und reale Adressen identisch.

z/OS benutzt dieses Verfahren. Der virtuelle Adressenraum eines Prozesses wird als (virtual) Address Space bezeichnet.

2.3 Betriebssystem Überwacher

2.3.1 System z Programmiermodell

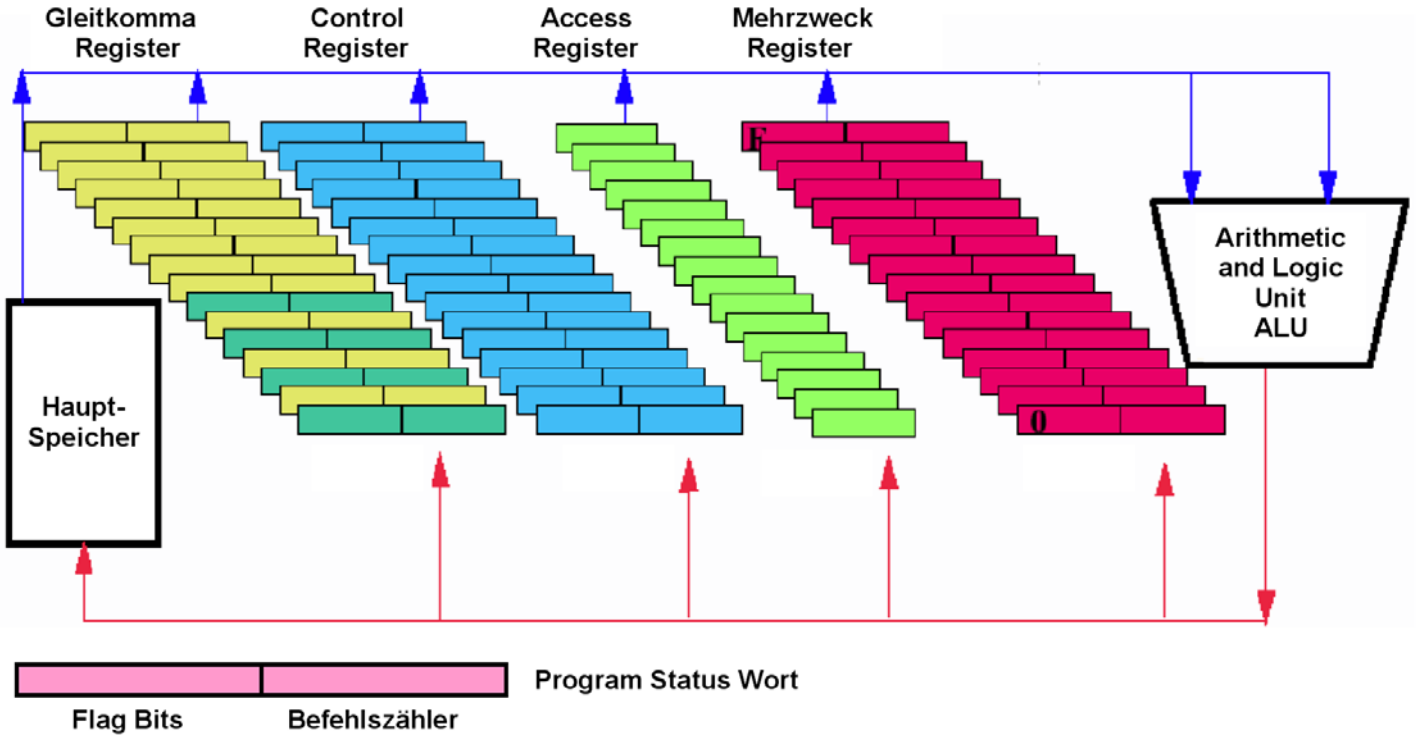

Abb. 2.3.1
Register, die dem Programmierer direkt oder indirekt zugänglich sind

Abbildung 2.3.1 zeigt die Bestandteile einer Mainframe CPU, die für den Programmierer sichtbar sind und die von Maschinenbefehlen manipuliert werden können. Man bezeichnet diese Teile auch als das Programmiermodell. Zu beachten ist, dass die CPU außerdem viele Teile und Funktionen enthält, die für einen Programmierer nicht zugänglich und unsichtbar sind.

Das Programmiermodell besteht aus dem Hauptspeicher, einer Verarbeitungseinheit (ALU) die Funktionen wie z.B. Addition ausführen kann und einer ganze Reihe von Registern, die Daten, Hauptspeicheradressen oder Steuerfunktionen speichern können. Der Bestand an System z Registern umfasst:

- **16 Mehrzweck Register (General Purpose Register, GPR, 64 Bit) speichern Adressen oder Daten.**
- **16 Gleitkommaregister (64 Bit) speichern Zahlen im Gleitkommaformat.**
- **16 Control Register (64 Bit) speichern Steuerfunktionen. Ein Beispiel ist Steuerregister 1, welches die Anfangsadresse der Seitentafel im Hauptspeicher enthält.**
- **16 Access Register (32 Bit) werden für spezielle Hauptspeicher Zugriffsfunktionen benutzt.**

Zusätzlich existiert ein 64 Bit Befehlszähler Register und ein Flag Bit Register. Der Befehlszähler enthält die Adresse des nächstens auszuführenden Maschinenbefehls. Das Flag Register enthält individuelle Bits. Ein Beispiel ist ein Bit, welches den Rechner entweder in den Benutzerstatus oder in den Überwacherstatus versetzt.

Als eine Besonderheit der System z Architektur werden Befehlzähler und Flag Bits zu einem einzigen 128 Bit langen Register, dem „Programm Status Wort“ (PSW) Register zusammengefasst. Vom Standpunkt der Ausführung eines Programms definiert der Inhalt des PSW in jedem Augenblick den Status der CPU.

Bei einem CPU Chip mit mehreren Cores enthält jedes Core einen eigenen Satz der hier gezeigten Register.

2.3.2 Überwacherstatus und Problemstatus

Eine CPU läuft in jedem Augenblick entweder im Überwacher Status (Supervisor State) oder im Benutzer Status (user State).

Der Überwacher Status bzw. Benutzer Status wird durch 1 Bit im Flag Bit Register Teil des Program Status Wortes der Zentraleinheit definiert.

Der Überwacher (Kernel) läuft im Überwacherstatus (Supervisor State, Kernel State).

Benutzerprogramme (Anwendungsprogramme) laufen im Problemstatus (Problem State, User State).

Auswirkungen sind:

- **Bestimmte „Privilegierte“ Maschinenbefehle können nur im Überwacherstatus ausgeführt werden**
- **Speicherschutz. Im Problemstatus kann nur auf einen Teil des virtuellen und des realen Hauptspeichers zugegriffen werden**

2.3.3 Unterbrechungen

Unterbrechungen (Interruptions) sind ein zentrales Steuerelement in einem jeden Rechner. Unterbrechungen bewirken eine Änderung des Status der Zentraleinheit als Folge von Bedingungen (Ursachen), die entweder

- **außerhalb der Zentraleinheit (CPU), oder**
- **innerhalb der Zentraleinheit**

auftreten.

Unterbrechungen bewirken den Aufruf und die Ausführung von speziellen Programmen (Unterbrechungsroutinen) außerhalb des normalen Programmablaufs.

Eine Unterbrechung bewirkt:

1. **Abspeichern des PSW (Programm-Status-Wort, Flag Bits und Befehlszählers) im Hauptspeicher.**
2. **Laden des Befehlszählers mit der Anfangsadresse einer Unterbrechungsroutine (Interrupt Handler).**
3. **Status-Initialisierung im PSW (z.B. Überwacherstatus setzen).**
4. **In der Regel: Abspeichern der Mehrzweckregister, evtl. auch Steuerregister, Gleitkommaregister durch die Unterbrechungsroutine.**

Typischerweise enthält ein Feld im Hauptspeicher zusätzliche Informationen über die Ursache der Unterbrechung.

Unterschiedliche Bedingungen können unterschiedliche Klassen von Unterbrechungen auslösen. Es existieren sechs Unterbrechungsklassen mit mehreren Unterbrechungsarten / Klasse

Maschinenfehler	**Beispiel Paritäts-Fehler (Bus, Hauptspeicher), fehlerhafte Addition**
Reset	**Beispiel: Setzt Zentraleinheit (und System) in jungfräulichen Zustand**
I/O	**Beispiel: Signalisiert den Abschluss einer E/A Operation**
Extern	**Beispiel: System externes Signal, z. B. Ablauf des Zeitgebers.**
Programm	**Beispiele: Division durch Null, Fehlseitenunterbrechung, illegaler OP Code, illegale Adresse**
Systemaufruf	**(SVC) Aufruf des Überwachers im Benutzerprogramm (eigentlich ein Maschinenbefehl)**

2.3.4 Unterbrechungsverarbeitung

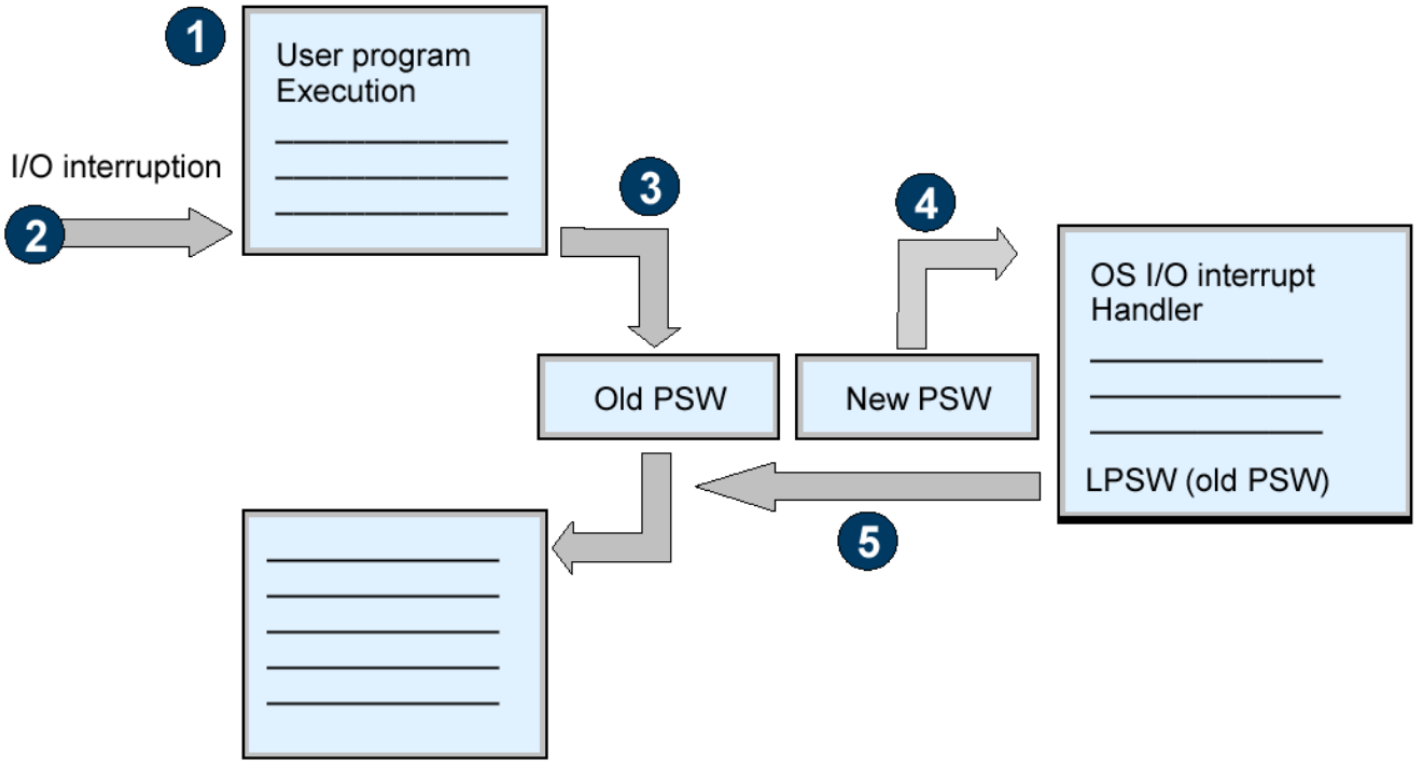

Abb. 2.3.2
Verarbeitung einer Unterbrechung

Eine Unterbrechung bewirkt das Abspeichern des derzeitig gültigen PSW (als old PSW bezeichnet) im Hauptspeicher an einer hierfür vorgesehenen Adresse und ein Laden eines neuen PSW (new PSW) in das PSW Register. Der Befehlszählerteil des neuen PSW enthält die Adresse des ersten Befehls der Unterbrechungsroutine. Die CPU befindet sich automatisch im Überwacherstatus.

Als allerletzten Befehl führt die Unterbrechungsroutine den Maschinenbefehl „LPSW (Load Programm Status) aus. Dieser bewirkt, dass das old PSW in das PSW Register geladen wird. Der Befehlszählerteil des PSW enthält die Adresse des Maschinenbefehls, den die CPU ohne Eintreten der Unterbrechung als nächstes ausgeführt hätte.

2.3.5 Schichtenmodell der Rechnerarchitektur

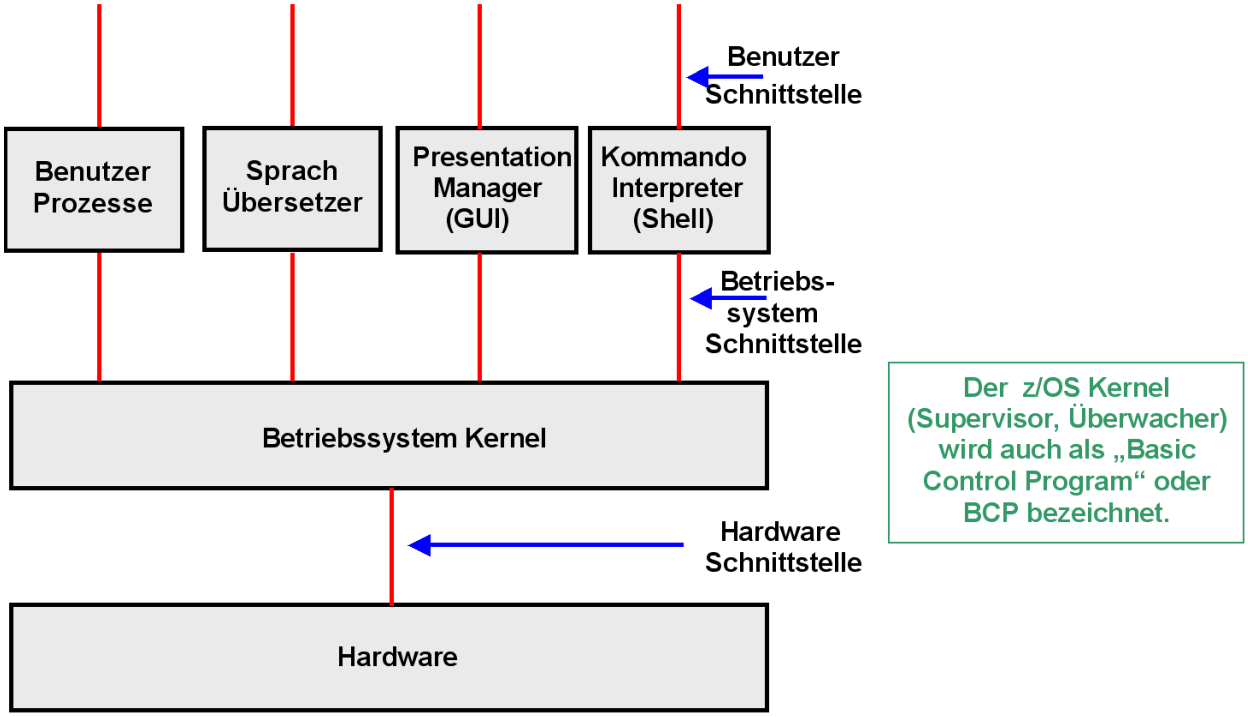

Abb. 2.3.3
Schichtenmodell der Rechnerarchitektur

Das Betriebssystem besteht aus einer zentralen Steuerungs- und Verwaltungsfunktion, dem Überwacher (Supervisor, Kernel), und weiteren Systemprogrammen oder Komponenten, die unter z/OS als „Subsysteme“ bezeichnet werden. Beispiele für Subsysteme sind der Kommandointerpreter (TSO), das Job Entry Subsystem(JES) oder das DB2 Datenbanksystem. Subsysteme sind Systemprozesse, die parallel zu den Benutzerprozessen laufen.

Der Überwacher (Supervisor) besteht aus :

1. **Datenbereichen (Control Blocks)**
2. **Programmteilen, welche Controlblock Daten manipulieren**

Der Überwacher ist In der Regel nicht strukturiert. In seinem Buch “Modern Operating Systems” published in 1992, pp.18, bezeichnet Prof. Andrew s. *Tanenbaum den Überwacher eines Betriebssystems als „.* The Big Mess“.

Der Überwacher enthält Funktionen die von vielen Prozessen gemeinsam genutzt werden. Häufig verbringt ein Prozess 50 % der Ausführungszeit (Pfadlänge) mit der Ausführung von Überwacherfunktionen (läuft 50 % der Zeit im Überwacherstatus).

Die Hardware des Rechners reagiert auf die Eingabe von Maschinenbefehlen und auf Unterbrechungen. Der Überwacher kann nur über Unterbrechungen aufgerufen werden. Er läuft im Überwacherstatus.

System - Aufrufe (System Calls) sind die einzige Möglichkeit für Benutzerprozesse, mit dem Überwacher zu kommunizieren. Unter z/OS implementiert der SVC Maschinenbefehl die System Call Funktion. Beim x86 hat der INT Maschinenbefehl die gleiche Funktion. Ein System Call ist eine Routine, die im Benutzer Status läuft, u.a. den SVC (oder INT) Maschinenbefehl ausführt, und dadurch den Übergang vom Benutzerstatus in den Überwacherstatus herbeiführt.

2.3.6 Systemaufruf (System Call)

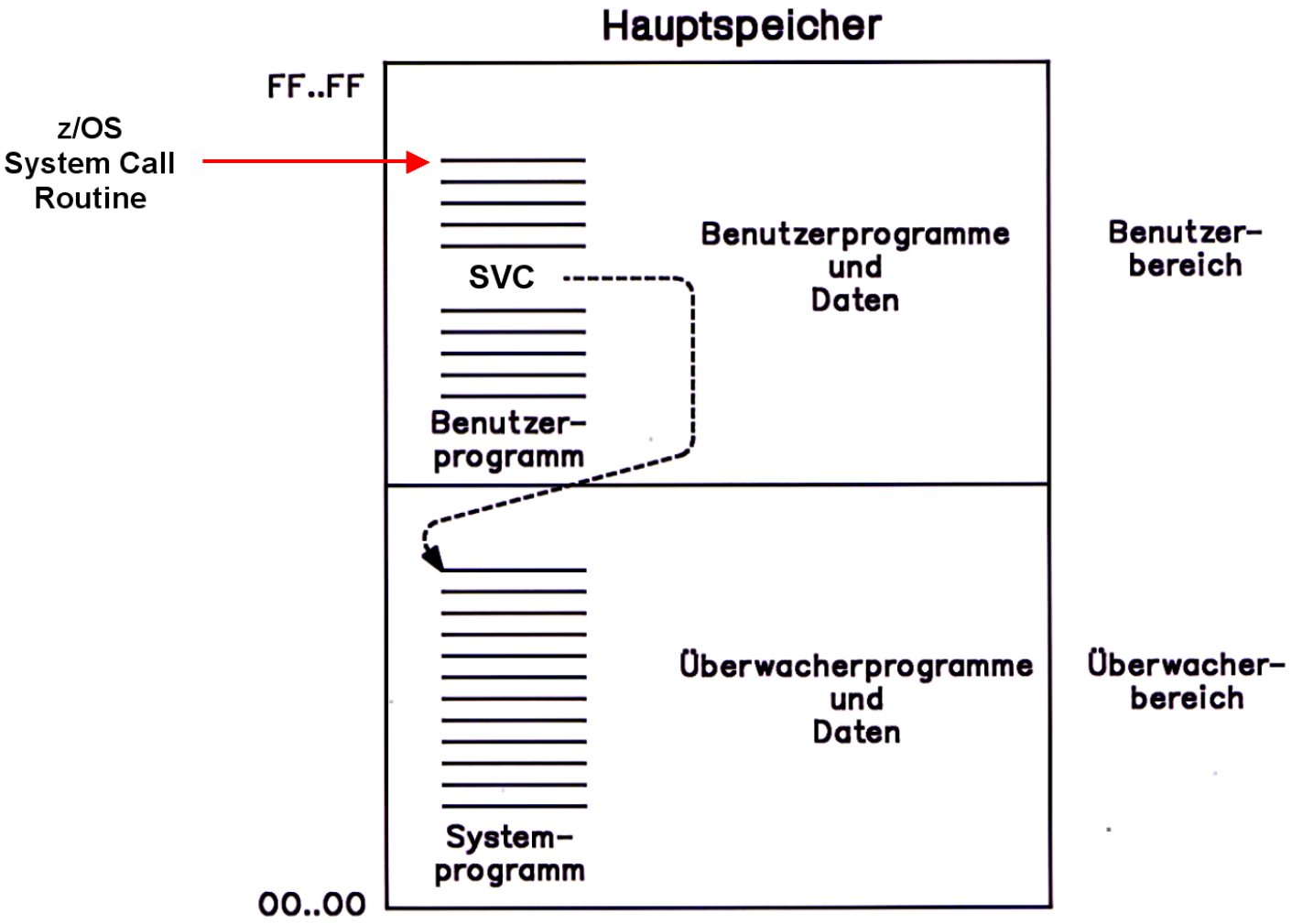

Abb. 2.3.4
Supervisor Call

Ein System z Benutzerprogramm kann eine Funktion des Überwachers durch Ausführung des SVC (Supervisor Call) Maschinenbefehls aufrufen. Der SVC Befehl übergibt hierzu einen Parameter, welcher die Art der gewünschten Funktion angibt. Ein Beispiel ist die Durchführung einer WRITE operation, welche Daten auf den Plattenspeicher schreibt. Die Ausführung des SVC Maschinenbefehls bewirkt eine Unterbrechung. Die Unterbrechungsroutine bewirkt unter anderem den Wechsel von Benutzerstatus in den Überwacher Status (Kernel Status).

2.3.7 Struktur des Supervisors

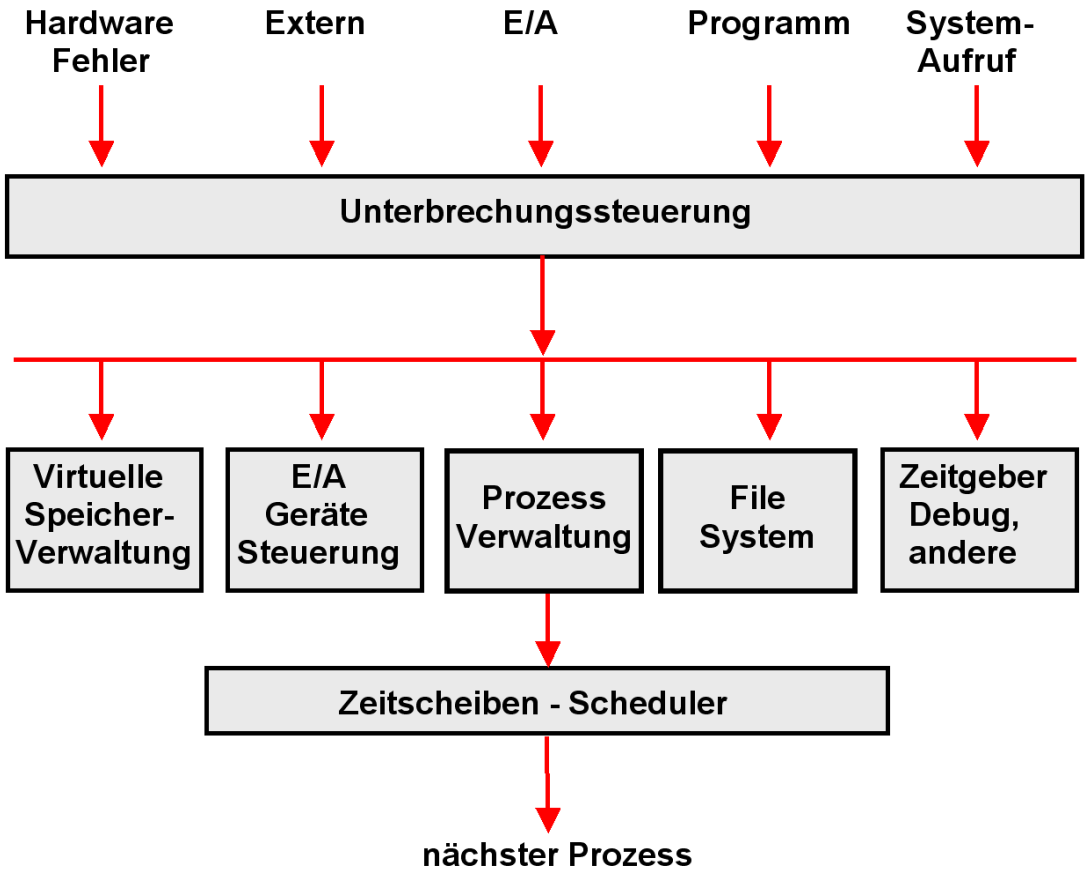

Abb. 2.3.5
Die wichtigsten Komponenten des Überwachers

Der Aufruf des Überwachers (Supervisor, Kernel) erfolgt grundsätzlich über Unterbrechungen.

Je nach Art der Unterbrechung werden unterschiedliche Komponenten des Überwachers aufgerufen.

Benutzerprozesse nehmen Dienste des Überwachers über eine architekturierte Schnittstelle, den Systemaufruf (System Call, Supervisor Call, SVC) in Anspruch. Diese Begriffe haben alle die gleiche Bedeutung.

Der Scheduler (Zeitscheibensteuerung) sucht den nächsten auszuführenden Prozess aus.

Der Überwacher wird über eine Unterbrechung aufgerufen. Dies ist der einzige Weg, über den man eine Funktion des Überwachers in Anspruch nehmen kann.

2.3.8 Funktionen des Überwachers

Die wichtigsten Überwacherfunktionen sind:

- **Die Unterbrechungssteuerung untersucht die Art der aufgetretenen Unterbrechung und ruft je nach Art eine andere Komponente des Überwachers auf.**
- **Die virtuelle Speicherverwaltung ordnet virtuellen und realen Speicherplatz zu. Sie bestimmt, welche Seiten in Rahmen des Hauptspeichers abgebildet werden und welche Seiten auf den externen Seitenspeicher (Auxiliary Store) ausgelagert werden. Sie lädt bei Bedarf Seiten vom externen Seitenspeicher in den Hauptspeicher.**
- **Die Input/Output Steuerung (I/O Supervisor) wird aufgerufen wenn ein Benutzerprogramm eine Lese oder Schreiboperation auf ein I/O Gerät (z.B. Festplattenspeicher) durchführen will. Sie nimmt auch eine I/O Unterbrechung entgegen, die z.B. besagt, dass ein Plattenspeicher eine I/O Operation erfolgreich abgeschlossen hat.**
- **Die Prozessverwaltung aktiviert und deaktiviert Prozesse. Aktive Prozesse verfügen über Ressourcen im Hauptspeicher. Bei deaktivierten Prozessen sind alle Resourcen (vermutlich) auf einen Plattenspeicher ausgelagert.**
- **Datenstrukturen auf Plattenspeichern werden mit Hilfe des File Systems verwaltet. Windows verwendet die NTFS und FAT32 File Systems. VSAM, PDSe und zFS sind die wichtigsten z/OS File Systems.**
- **Zahlreiche weitere Funktionen wie Zeitscheibensteuerung und Funktionen zum debuggen von Software sind vorhanden.**
- **Der Scheduler verwaltet die Warteschlangen der TCBs. Er versetzt Prozesse in den Zustand wartend, ausführbar oder laufend. Er steuert Prioritäten, nach denen entschieden wird, welcher Prozess vom Zustand wartend in den Zustand ausführbar überführt wird.**

Ein Prozess wird durch seinen TCB repräsentiert und dargestellt. Der TCB ist ein Bereich im Hauptspeicher und enthält Informationen über den Prozess.

Mehrzweck Register , Gleitkommaregister usw. sowie das PSW sind in einer CPU nur einmal vorhanden. Wird ein Prozess vom laufenden in den wartenden Zustand versetzt, wird der Inhalt der Register und des PSW in seinem TCB abgespeichert, ebenso alle weitere Information, die den Prozess charakterisiert. Wird der gleiche Prozess später einmal wieder vom ausführbaren in den laufenden Zustand versetzt, wird diese Information benutzt, um den bisherigen Zustand der CPU wieder herzustellen, indem der Inhalt der Register und des PSW mittels der im TCB gespeicherten Information wieder in den alten Zustand versetzt wird..

2.3.9 TCB Queues

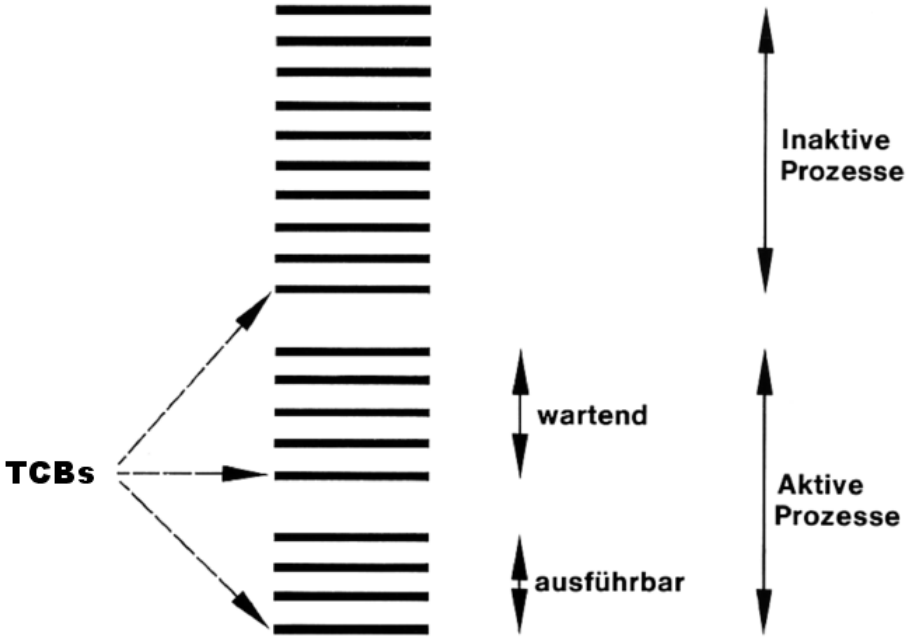

Abb. 2.3.6
Warteschlange der Task Control Blöcke

Ein Prozess wird durch einen Prozessleitblock beschrieben. Der Prozessleitblock enthält wichtige Funktionen für die Ausführung eines Prozesses.

z/OS verwendet die Bezeichnung Task Control Block (TCB). Andere Architekturen verwenden die Bezeichnung Process Control Block (PCB).

Zu jedem Zeitpunkt beanspruchen viele Prozesse Platz im Hauptspeicher und könnten ausgeführt werden. In einem Rechner mit einer CPU verarbeitet die CPU aber nur einen Prozess. In einem Mehrfachrechner mit n CPUs können gleichzeitig n Prozesse ausgeführt werden.

TCBs der wartenden und ausführbaren Prozesse werden vom Scheduler in Warteschlangen eingereiht und verwaltet.

Prozesse können inaktiv sein. Dies bedeutet, ihre TCBs und ihre Seiten sind auf einen Plattenspeicher ausgelagert. Der Scheduler entscheidet, ob und wann ein inaktiver Prozess aktiviert wird.

Ein Prozess wird durch seinen TCB repräsentiert und dargestellt. Der TCB ist ein Bereich im Hauptspeicher und enthält Informationen über den Prozess.

Mehrzweck Register , Gleitkommaregister usw. sowie das PSW sind in einer CPU nur einmal vorhanden. Wird ein Prozess vom laufenden in den wartenden Zustand versetzt, wird der Inhalt der Register und des PSW in seinem TCB abgespeichert, ebenso alle weitere Information, die den Prozess charakterisiert. Wird der gleiche Prozess später einmal wieder vom ausführbaren in den laufenden Zustand versetzt, wird diese Information benutzt, um den bisherigen Zustand der CPU wieder herzustellen.

2.3.10 Scheduler

Der Scheduler ist die Komponente des Überwachers, welcher drei Warteschlangen für die laufenden, wartenden und ausführbaren TCBs verwaltet. Er überführt bei gegebenen Anlass einen TCB von einer Warteschlange in eine andere.

Wenn eine CPU frei wird, selektiert der Scheduler aus der Warteschlange der ausführbaren TCBs einen Prozess und versetzt ihn in den Zustand laufend. Dies kann über Prioritäten gesteuert werden, und der z/OS Überwacher verfügt hierzu über eine Prioritätssteuerung.

In der Mehrzahl der Fälle bleibt ein Prozess nur kurze Zeit im Zustand laufend. Um zu verhindern, dass ein bestimmter Prozess eine CPU zu lange in Anspruch nimmt, verfügt der Scheduler über eine Zeitscheibensteuerung. Eine Zeitscheibe ist ein Zeitintervall von typischerweise wenigen Millisekunden. Verbleibt ein Prozess im Zustand laufend über seine Zeitscheibenlänge hinaus, erfolgt eine (Seitenfehler, page fault) Unterbrechung durch einen Zeitgeber. Diese bewirkt, dass der laufende Prozess in den Zustand ausführbar versetzt wird, und ein anderer Prozess dafür in den Zustand laufend versetzt wird.

In der Regel laufen unterschiedliche Prozesse auf den CPUs eines Rechners. Es ist jedoch möglich, dass ein Prozess mehr als eine CPU in Anspruch nimmt.

2.4 Cache

2.4.1 Halbleiter Speicher Chips

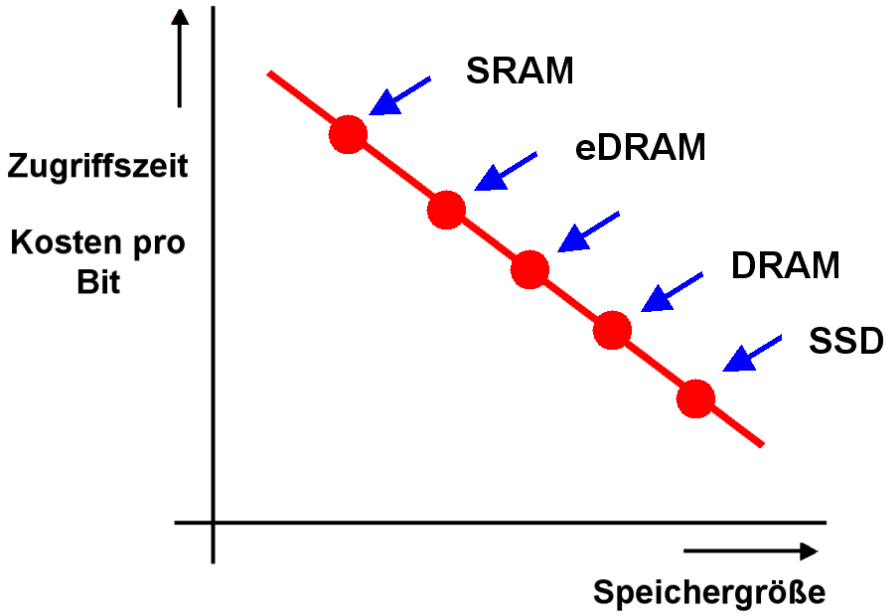

Abb. 2.4.1
Relation Zugriffszeit und Speichergröße

Es existieren eine ganze Reihe unterschiedlicher Technologien, mit denen man einen Speicher auf einem Halbleiterchip realisieren kann. Die Technologien unterscheiden sich in der Zugriffszeit auf den Speicher sowie der Speicherdichte, der Anzahl der Bits, die man pro mm^2 unterbringen kann. Die Speicherdichte beeinflusst die Kosten pro Bit. Allgemein gilt der hier dargestellte Zusammenhang: Kleine Speicher (Schnellspeicher) haben eine schnelle Zugriffszeit, brauchen viel Platz pro Bit und haben hohe Kosten pro Bit. Große Speicher haben eine längere Zugriffszeit, brauchen wenig Platz pro Bit und haben niedrige Kosten pro Bit.

Hauptspeicher Chips verwenden die DRAM (Dynamic Random Access Memory) Technologie. Das speichernde Element ist dabei ein Kondensator, der entweder geladen oder entladen ist. Über einen Schalttransistor wird er zugänglich und entweder ausgelesen oder mit neuem Inhalt beschrieben. Der Speicherinhalt ist (wie auch bei SRAM und eDRAM Chips) flüchtig (volatil), das heißt, die gespeicherte Information geht bei fehlender Stromversorgung oder zu später Wiederauffrischung verloren.

Festplattenspeicher sind dagegen statisch. Die gespeicherten Daten bleibt auch im abgeschalteten Zustand erhalten.

2.4 2 Cache Speicher

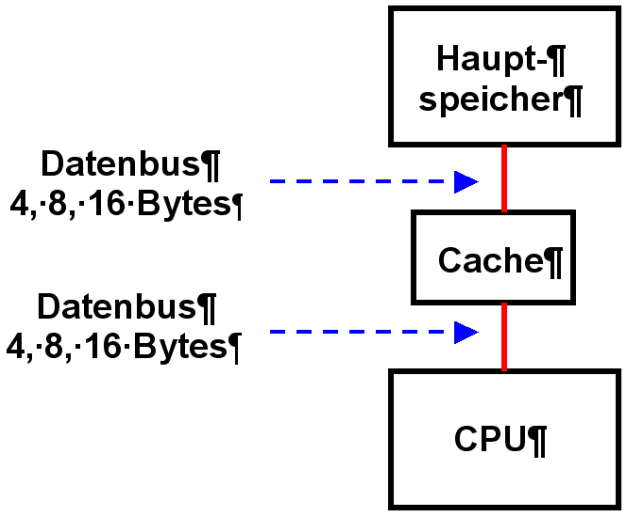

Abb. 2.4.2
Verbesserung der Hauptspeicher Zugriffszeit

Hauptspeicher Zugriffszeiten sind zu langsam, um mit der heute möglichen Verarbeitungsgeschwindigkeit einer CPU mithalten zu können. Deswegen schaltet man zwischen Hauptspeicher und CPU einen Cache Speicher. Der Cache Speicher verwendet eine SRAM (Static Random Access Memory) Technologie. Das speichernde Element ist dabei ein FlipFlop. Es existieren viele unterschiedliche SRAM Technologien, mit unterschiedlichen Zugriffszeiten, Speicherdichten und Kosten. Während mit DRAMs aufgebaute Hauptspeicher eine Zugriffszeit in der Gegend von 100 ns aufweisen, haben mit SRAMs aufgebaute Cache Speicher Zugriffszeiten zwischen 100 ps und 10 ns .

Wenn man von einem Cache redet meint man meistens einen Hauptspeicher Cache. Da es auch Plattenspeicher-Caches und andere Caches gibt ist die Unterscheidung wichtig.

Jeder moderne Rechner hat einen oder mehrere Caches. Die Existenz des Caches ist für den Benutzer praktisch unsichtbar. Es existieren auch keine Maschinenbefehle, mit denen der Programmierer den Inhalt des Caches manipulieren kann.

2.4.3 Struktur des Cache Speichers

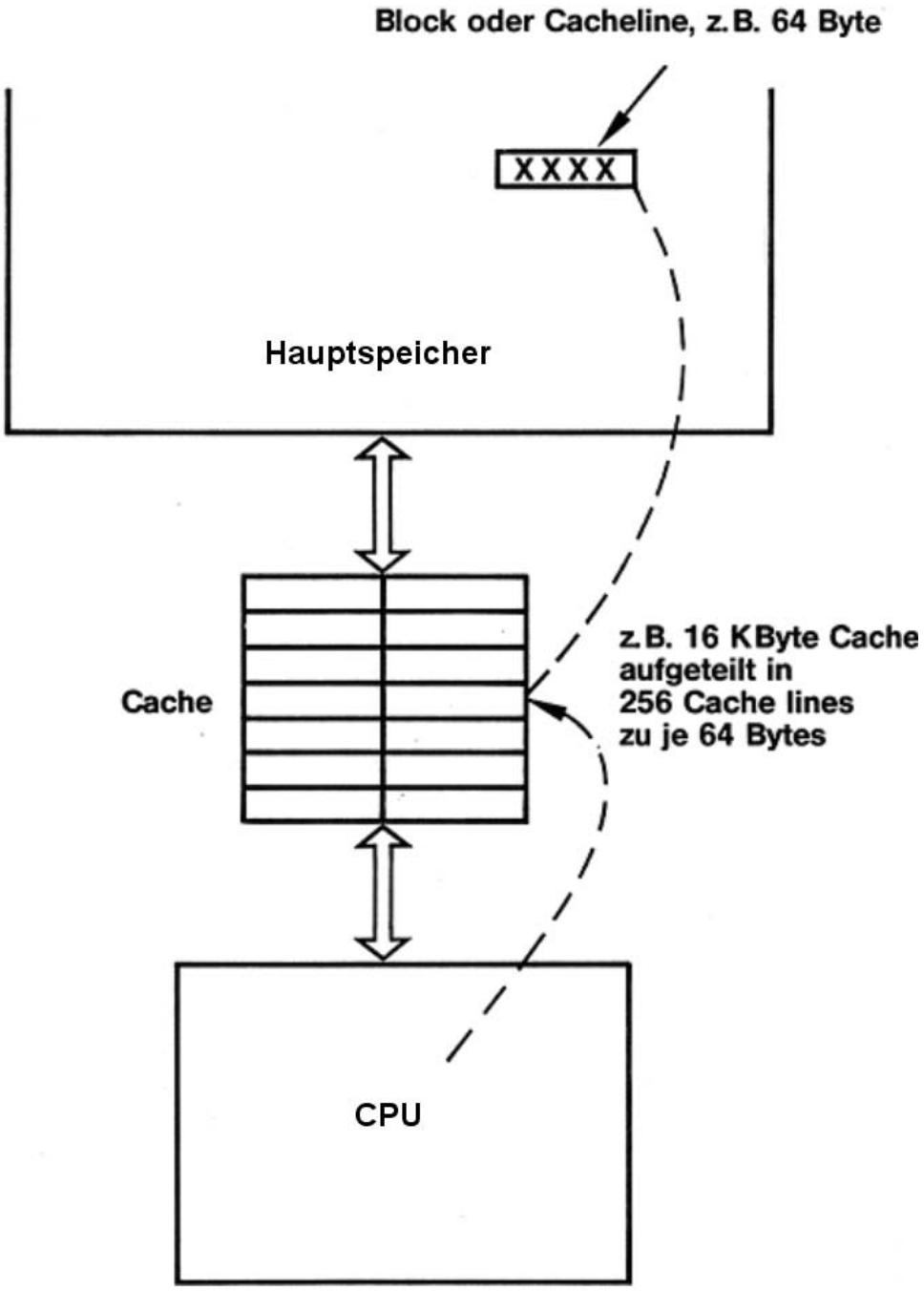

Abb. 2.4.3
Aufteilung des Cache Speichers in Cache Lines

Der Cache enthält in jedem Augenblick nur die Kopie einer Untermenge der Daten im Hauptspeicher. Diese Untermenge wird ständig ausgewechselt.

Hierzu werden Hauptspeicher und Cache Speicher in Blöcke mit einer identischen Größe aufgeteilt. Diese Blöcke werden als „Cachelines“ bezeichnet. Die Größe der Cachelines ist implementierungs-abhängig. Bei den z10 und z196 Mainframes sind es 256 Bytes, bei anderen Rechnern häufig weniger.

Ein Prozessor mit einer Cacheline-Size von 256 Byte wird aus dem Hauptspeicher immer nur Pakete dieser Größe in den Cache transportieren. Bei einem Hauptspeicher-Interface mit z.B. 256 Datenleitungen bedeutet das, dass jeder Cache Miss (wenn also angeforderte Daten nicht im Cache stehen) eine Adressierung und 8 Daten-Transferzyklen nach sich zieht.

2.4.4 Cache Directory

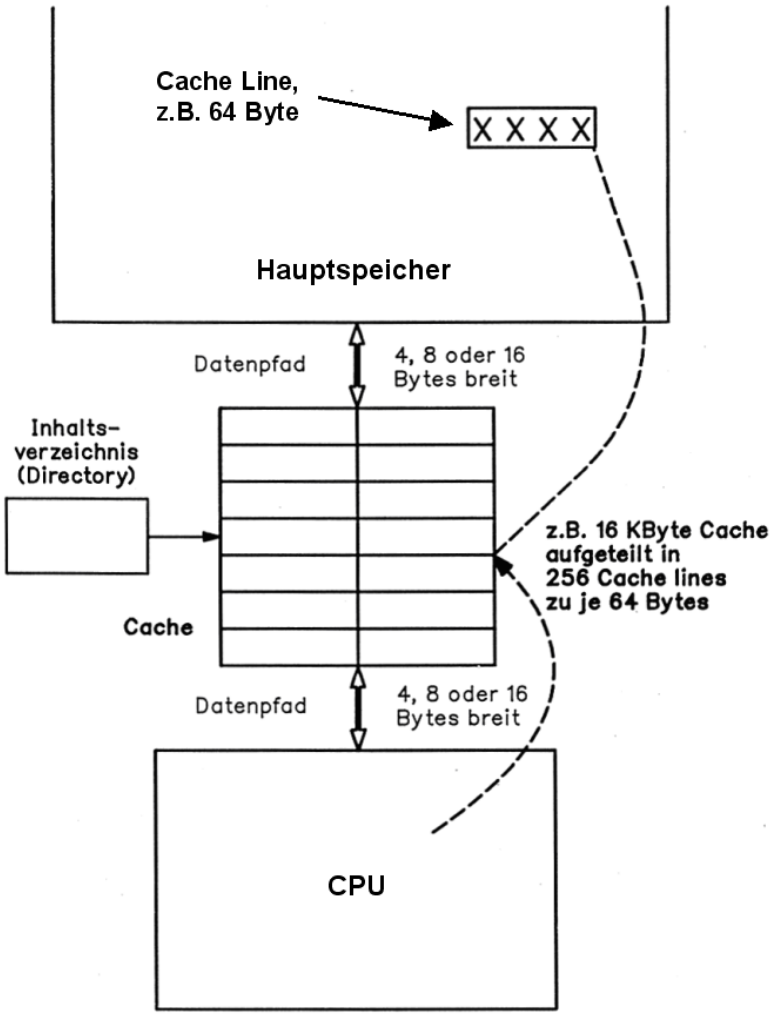

Abb. 2.4.4
Adressierung des Cache Speichers mittels des Cache Directories

Die Anordnung der Cachelines ist willkürlich und ändert sich ständig. Die CPU konsultiert ein „Cache Directory“ (Inhaltsverzeichnis) um eine bestimmte Cacheline innerhalb des Caches zu finden.

Das Cache Directory enthält je einen Eintrag für jede im Cache gespeicherte Cache Line. Der Eintrag enthält die Adresse der Cacheline im Hauptspeicher und im Cache.

Bei jedem Hauptspeicherzugriff durchsucht die CPU das Cache Directory in der Hoffnung, dass die benötigte Cacheline sich im Cache befindet. Wenn das nicht der Fall ist entsteht ein Cache Miss. Dies bewirkt, dass die benötigte Cache Line in ihrer Gesamtheit vom Hauptspeicher in den Cache geladen wird. Vermutlich muss dabei eine andere (nicht mehr benötigte) Cache Line aus dem Cache ausgelagert werden um Platz zu schaffen.

2.4.5 Split Cache

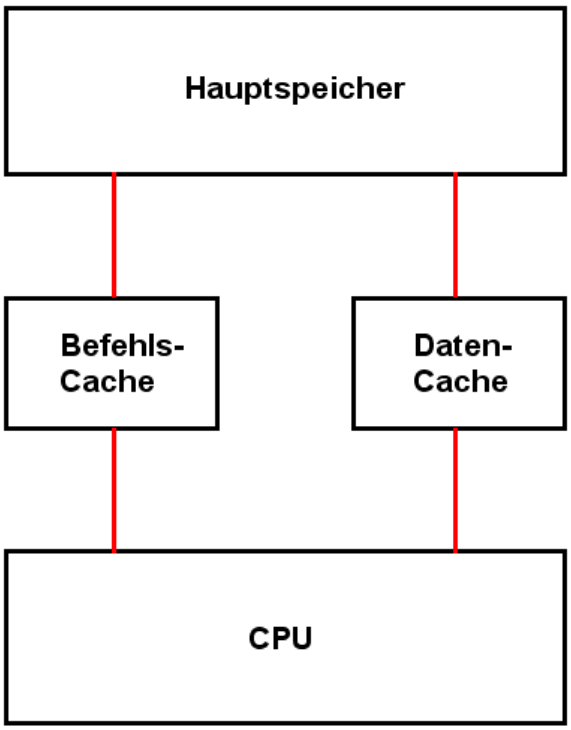

Abb. 2.4.5
Maschinen Befehle und Operanden in getrennten Cache Speichern

Im Hauptspeicher sind Programme (bestehend aus Maschinenbefehlen) und Daten gespeichert. Diese befinden sich in unterschiedlichen Hauptspeicherbereichen und damit auch in unterschiedlichen Cachelines.

Um den Durchsatz zu verbessern besteht der Cache häufig aus zwei unabhängigen Cache Speichern (Split Cache, Dual Cache): Der Befehlscache enthält nur Maschinenbefehle und der Datencache enthält nur Daten.

Der z9, z10, z196 und zEC12 Cache ist ein Dual Cache.

Der Befehlscache wird häufig als I-Cache (Instruction Cache) und der Datencache als D-Cache bezeichnet.

2.4.6 Second Level Cache

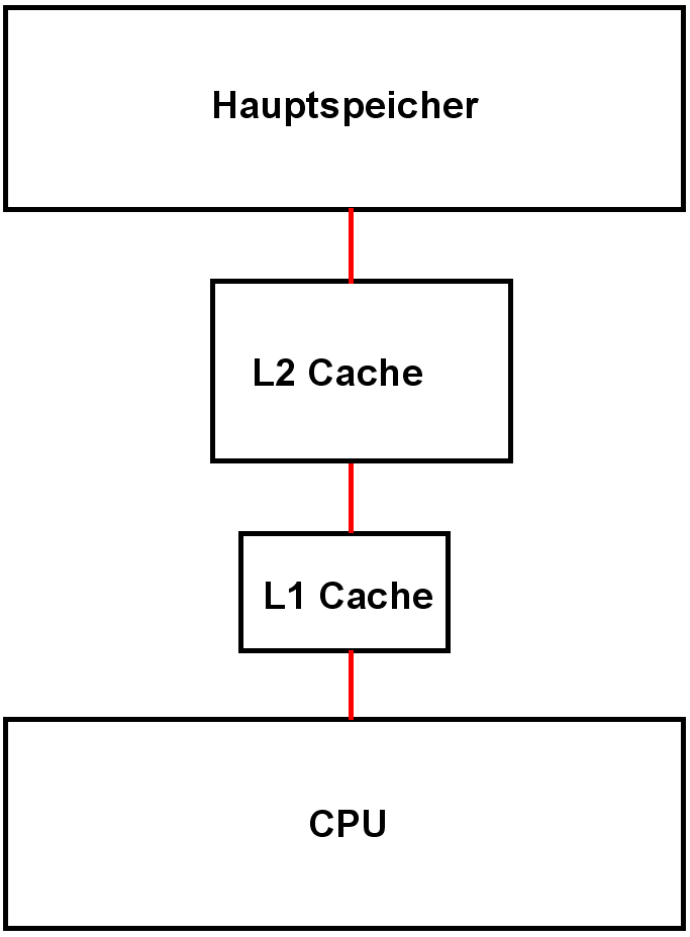

Abb. 2.4.6
SRAM Technologie mit unterschiedlichen Zugriffszeiten

Es existieren zahlreiche SRAM Technologien um Caches zu implementieren. Heute ist es üblich, mit einer Cache Hierarchie zu arbeiten. Hierbei wird ein „Level 1“ Cache (L1) mit besonders schnellen, aber teuren und platzaufwendigen SRAM Speicherzellen implementiert. Ein „Level 2“ Cache (L2) verwendet langsamere aber dafür kostengünstigere SRAM Zellen.

Hierbei wird häufig, z.B. beim z9 Mainframe, der L1 Cache als Split Cache und der L2 Cache als Uniform Cache implementiert

2.4.7 Mainframe Cache Hierarchien

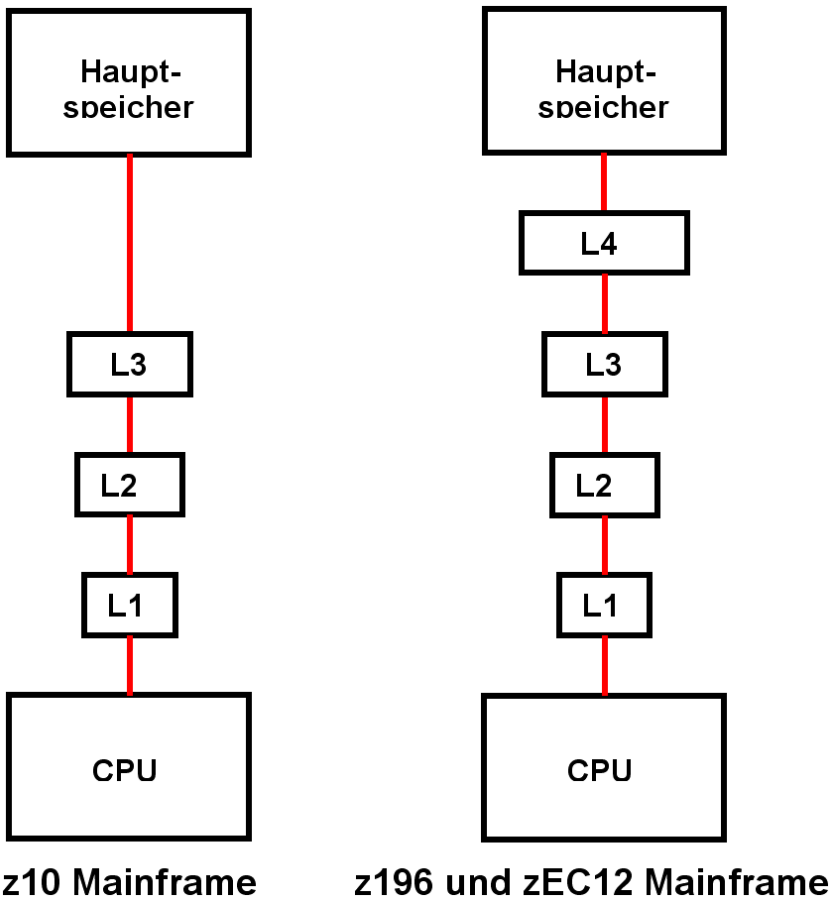

Abb. 2.4.7
Die L3 und L4 Stufen benutzen die langsamere eDRAM Technologie

Moderne Cache Hierarchien sind noch komplizierter. Gezeigt sind die 3-stufige z10 Cache Hierarchie und die 4-stufige z196 und zEC12 Cache Hierarchie. Sinnvoll wurde dies durch die Erfindung einer neuartigen als eDRAM bezeichneten Cache Speicherzellen-Technologie, die im z196 und zEC12 Mainframe die SRAM Zellen in den L3 und L4 Caches ersetzt.

2.5 Weiterführende Information

Die hier folgende Präsentation ist kein Prüfungsstoff.

Aber vielleicht interessiert Sie, wie sich die Rechentechnologie im Laufe der Jahrtausende entwickelt hat.

Sie können die Entwicklung verfolgen unter

http://www.cedix.de/VorlesMirror/Band1/History.pdf

Das folgende Video zeigt eine Demonstration des Antikythera Computers:
http://www.cedix.de/VorlesMirror/Band1/Antikythera.html
Vorsicht: der Download umfasst 67 MByte - bitte warten. Das Original ist zu finden unter:

http://www.cedix.de/VorlesMirror/Band1/Antikythera-M4-DivxQ85.avi