Systemprogrammierung

Grundlagen von Betriebssystemen

Teil C – XII.1 Speicherverwaltung: Adressräume

Wolfgang Schröder-Preikschat

20. Dezember 2019

Agenda

Einführung
- Rekapitulation

Adressräume
- Real
- Logisch
- Virtuell

Mehradressraumsysteme
- Virtualität
- Exklusion
- Inklusion

Zusammenfassung

Gliederung

Einführung
- Rekapitulation

Adressräume
- Real
- Logisch
- Virtuell

Mehradressraumsysteme
- Virtualität
- Exklusion
- Inklusion

Zusammenfassung

Lehrstoff

- **Adressräume** detailiert behandeln, um Bedeutungen der jeweiligen Ausprägungen erfassen zu können
  - real: manifestiert im **Adressraumbelegungsplan** des Herstellers
  - logisch: abstrahiert von diesem Plan, aber nicht vom Speicher
  - virtuell: abstrahiert von der Speicherlokalität (Vorder-/Hintergrund)
- gängige **Adressumsetzungstechniken** vorstellen und vertiefen, um die logische/virtuelle Adresse auf eine reale abbilden zu können
  - seitennummeriert/gekachelt, Seitentabelle und -deskriptor
  - explizit oder implizit segmentiert, Segmenttabelle und -deskriptor
  - segmentiert und seitennummeriert, als Kombination beider Techniken
  - inkl. **Übersetzungspuffer** zur Latenzverbergung bei der Abbildung
- Modelle von **Mehradressraumsysteme** kennenlernen und sie in ihren nichtfunktionalen Eigenschaften differenzieren können
  - Exklusion: total private Adressräume für alle Programme
  - Inklusion: partiell private Adressräume für Maschinenprogramme

[5, S. 9]

Referenzfolge

reference string

- ein **„laufender“ Prozess** [4, S. 19] generiert Folgen von Adressen auf den Haupt-/Arbeitsspeicher, und zwar:
  - i nach Vorschrift des Programms, das diesen Prozess spezifiziert, wie auch
  - ii in Abhängigkeit von den Eingabedaten für den Programmablauf
- der **Wertevorrat** dieser Adressen ist aber stets gemäß Programm in der Größe nach oben begrenzt
  - er ist initial statisch und gibt die zur Programmausführung mindestens erforderliche Menge an Haupt-/Arbeitsspeicher vor
  - jedoch gestaltet er sich zur Laufzeit dynamisch, nimmt zu und kann dabei aber den „einem Prozess zugebilligten“ Wertevorrat nicht überschreiten
  - letzteres sichert der Kompilierer (typsichere Programmiersprache) oder das Betriebssystem (in Zusammenspiel mit der MMU) zu
- der einem Prozess zugebilligte Wertevorrat gibt den **Adressraum** vor, in dem dieser Prozess (logisch/physisch) eingeschlossen ist
  - der Prozess kann aus seinem Adressraum normalerweise nicht ausbrechen und folglich nicht in fremde Adressräume eindringen
  - der **Prozessadressraum** hat eine maximale, hardwarebeschränkte Größe

[5, S. 12]

Adressraumlehre I

totale Abbildung $f : A_l \rightarrow A_r$

- Befehlssatzebene (Ebene$_2$)

Definition (realer Adressraum)

Der durch einen Prozessor definierte Wertevorrat $A_r = [0, 2^n - 1]$ von Adressen, mit $e \leq n \leq 64$ und (norm.) $e \geq 16$. Nicht jede Adresse in A_r ist jedoch gültig, d.h., A_r kann Lücken aufweisen.

  - der **Hauptspeicher** ist adressierbar durch einen oder mehrere Bereiche in A_r, je nach Hardwarekonfiguration
- Maschinenprogrammebene (Ebene$_3$)

Definition (logischer Adressraum)

Der in Programm P definierte Wertevorrat $A_l = [n, m]$ von Adressen, mit $A_l \subset A_r$, der einem Prozess von P zugebilligt wird. Jede Adresse in A_l ist gültig, d.h., A_l enthält *konzeptionell* keine Lücken.

  - führt **Arbeitsspeicher** ein, der linear adressierbar ausgelegt ist und durch das Betriebssystem auf den Hauptspeicher abgebildet wird

Adressraumlehre II

partielle Abbildung $f : A_v \rightsquigarrow A_r$

- Maschinenprogrammebene (Ebene$_3$)

Definition (virtueller Adressraum)

$A_v = A_l$: A_v übernimmt alle Eigenschaften von A_l. Jedoch nicht jede Adresse in A_v bildet ab auf ein im Hauptspeicher liegendes Datum.

  - Benutzung einer solchen nicht abgebildeten Adresse in A_v verursacht in dem betreffenden Prozess einen **Zugriffsfehler**
  - der Prozess erfährt eine **synchrone Programmunterbrechung** (*trap*), die vom Betriebssystem behandelt wird
  - das Betriebssystem sorgt für die **Einlagerung** des adressierten Datums in den Hauptspeicher und
  - der Prozess wird zur **Wiederholung** der gescheiterten Aktion gebracht
- der durch A_v für den jeweiligen Prozess benötigte Hauptspeicher ist „nicht in Wirklichkeit vorhanden, aber echt erscheinend“
  - jedoch steht jederzeit genügend Arbeitsspeicher für A_v zur Verfügung
    - einesteils im Hauptspeicher, anderenteils im Ablagespeicher (*swap area*)

Gliederung

Einführung
Rekapitulation

Adressräume
Real
Logisch
Virtuell

Mehradressraumsysteme
Virtualität
Exklusion
Inklusion

Zusammenfassung

Adressraumbelegung

address-space assignment

- ein **Adressraumbelegungsplan** bestimmt, welche Hardwareeinheiten über welche Adressen oder Adressbereiche zugreifbar sind
  - nicht nur **Speicher** (RAM, ROM), sondern auch **Peripheriegeräte**

Adressbereich	Größe (KiB)	Verwendung
00000000–0009ffff	640	RAM (System)
000a0000–000bffff	128	Video RAM
000c0000–000c7fff	32	BIOS Video RAM
000c8000–000dffff	96	keine
000e0000–000effff	64	BIOS Video RAM (*shadow*)
000f0000–000fffff	64	BIOS RAM (*shadow*)
00100000–090fffff	147456	RAM (Erweiterung)
09100000–fffdffff	4045696	keine
fffe0000–fffeffff	64	SM-RAM (*system management*)
ffff0000–ffffffff	64	BIOS ROM

Toshiba Tecra 730CDT, 1996

- die konkrete Auslegung gibt der **Hersteller des Rechensystems** vor, nicht der Hersteller (Intel) des Prozessors

Ungültige Adressen

- im **Adressraumbelegungsplan** finden sich auch Adressbereiche, mit denen keine Hardwareeinheiten assoziiert sind
  - der Zugriff darauf ist undefiniert oder liefert einen **Busfehler** (*bus error*)

Adressbereich	Größe (KiB)	Verwendung
00000000–0009ffff	640	RAM (System)
000a0000–000bffff	128	Video RAM
000c0000–000c7fff	32	BIOS Video RAM
000c8000–000dffff	96	keine
000e0000–000effff	64	BIOS Video RAM (*shadow*)
000f0000–000fffff	64	BIOS RAM (*shadow*)
00100000–090fffff	147456	RAM (Erweiterung)
09100000–fffdffff	4045696	keine
fffe0000–fffeffff	64	SM-RAM (*system management*)
ffff0000–ffffffff	64	BIOS ROM

Toshiba Tecra 730CDT, 1996

- **zeitgenössische Hardware** lässt den Zugriff mit ungültigen Adressen nicht undefiniert, sondern unterbricht den zugreifenden Prozess (*trap*)

Reservierte Adressen

- aber auch Adressbereiche mit speziellem Verwendungszweck sind im **Adressraumbelegungsplan** festgeschrieben
  - der Zugriff darauf bedeutet eine **Zugriffsverletzung** (*access violation*)

Adressbereich	Größe (KiB)	Verwendung
00000000–0009ffff	640	RAM (System)
000a0000–000bffff	128	Video RAM
000c0000–000c7fff	32	BIOS Video RAM
000c8000–000dffff	96	keine
000e0000–000effff	64	BIOS Video RAM (*shadow*)
000f0000–000fffff	64	BIOS RAM (*shadow*)
00100000–090fffff	147456	RAM (Erweiterung)
09100000–fffdffff	4045696	keine
fffe0000–fffeffff	64	SM-RAM (*system management*)
ffff0000–ffffffff	64	BIOS ROM

Toshiba Tecra 730CDT, 1996

- typischerweise sind dies Adressbereiche, die der **residente Monitor** (BIOS, *Open Firmware* (OFW), EFI) sein Eigen nennt

Freie Adressen

- schließlich benennt der **Adressraumbelegungsplan** Adressbereiche, die dem allgemeinen Verwendungszweck unterliegen
  - der Zugriff darauf kann einen **Schutzfehler** (*protection fault*) liefern

Adressbereich	Größe (KiB)	Verwendung
00000000–0009ffff	640	RAM (System)
000a0000–000bffff	128	Video RAM
000c0000–000c7fff	32	BIOS Video RAM
000c8000–000dffff	96	keine
000e0000–000effff	64	BIOS Video RAM (*shadow*)
000f0000–000fffff	64	BIOS RAM (*shadow*)
00100000–090fffff	147456	RAM (Erweiterung)
09100000–fffdffff	4045696	keine
fffe0000–fffeffff	64	SM-RAM (*system management*)
ffff0000–ffffffff	64	BIOS ROM

Toshiba Tecra 730CDT, 1996

- der **Hauptspeicher** (*main memory*), in dem Betriebssystem und die Maschinenprogramme (in Gänze/Teilen) zeitweilig liegen

Abstraktion von der realen Adressraumbelegung

- ein **logischer Adressraum** beschreibt die geradlinige Beschaffenheit des Hauptspeichers eines (schwergewichtigen) Prozesses
  - Hauptspeicher getrennter realer Adressbereiche wird **linear adressierbar**
  - zuzüglich speicherabgebildeter (*memory mapped*) Entitäten der Hardware
    - hier insb. Geräteregister zur Interaktion mit Peripheriegeräten
    - d.h., zur speicherabgebildeten Ein-/Ausgabe (*memory-mapped I/O*)
- er umfasst alle für einen Prozess gültigen Text- und Datenadressen, entsprechend des durch ihn ablaufenden Programms
  - auf Programmiersprachenebene bezugnehmend auf mind. zwei Segmente
    - Text – Maschinenanweisungen, Programmkonstanten
    - Daten – initialisierte Daten, globale Variablen, Halde
  - auf Maschinenprogrammebene mindestens ein weiteres Segment
    - Stapel – lokale Variablen, Hilfsvariablen, aktuelle Parameter
    - andere – Gemeinschaftsbibliotheken (*shared libraries*) oder -einrichtungen
- festgelegt durch das **Adressraummodell** (S. 26) des Betriebssystems und von letzteres abgebildet auf den realen Adressraum
  - mittels MMU (*memory management unit*), die dazu eine gekachelte bzw. seitennummerierte oder segmentierte Organisationsstruktur definiert

Eindimensionaler Adressraum

gekachelt

Seitennummerierung *steht für eine Unterteilung des Adressraums in gleichgroße Einheiten und deren* ***lineare Aufzählung****.*

- je nach Adressraumtyp werden diese Einheiten verschieden benannt

  Seite (*page*) im logischen/virtuellen Adressraum

  Seitenrahmen (*page frame*), auch Kachel, im realen Adressraum
- die vom Prozess generierte lineare Adresse *la* ist ein Tupel (p, o):
  - p ist eine **Seitennummer** (*page number*) im Adressraum $[0, 2^N - 1]$
    - Wertebereich für $p = [0, (2^N \texttt{ div } 2^O) - 1]$
  - o ist der **Versatz** (*offset*, *displacement*) innerhalb von Seite p
    - Wertebereich für $o = [0, 2^O - 1]$

  ↪ mit $O << N$ und 2^O auch Seitengröße (in Bytes): typisch ist $2^{12} = 4096$
- tabellengesteuerte Abbildung von *la* mit p als **Seitenindex**
  - **Seitentabelle** (*page table*) von sogenannten **Seitendeskriptoren**
    - auch **Seiten-Kachel-Tabelle**, ein „dynamisches Feld“
    - wobei ggf. mehrere solcher Felder pro Prozessexemplar angelegt sind

Seitendeskriptor

page descriptor

- ein von der Hardware (MMU) vorgegebener **Verbund** von Daten, der statische/dynamische **Seiteneigenschaften** beschreibt:

  Kachel-/Seitenrahmennummer
  - seitenausgerichtete reale Adresse

  Attribute
  - Schreibschutzbit
  - Präsenzbit (an-/abwesend)
  - Referenzbit[1]
  - Modifikationsbit[1]

  - je nach Hardware und Adressraummodell gibt es weitere Attribute
    - Privilegstufe, Seiten(rahmen)größe, Spülungssteuerung (TLB), ...

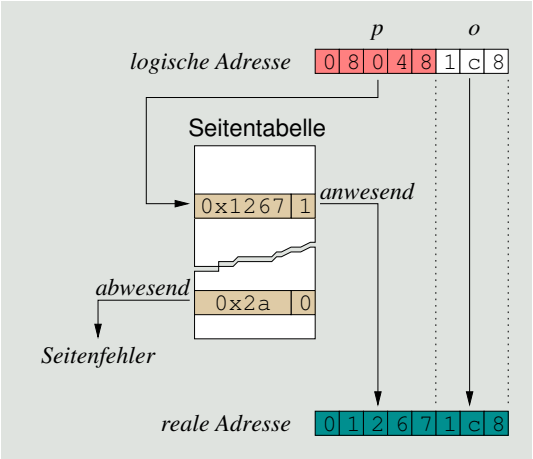
- Betriebssysteme definieren pro Seitendeskriptor oft Attribute, die im **Schatten** der Seitentabelle gehalten werden müssen
  - Seitendeskriptor des Betriebssystems in der „*shadow page table*"
  - für allgemeine Verwaltungsaufgaben, aber auch speziellen Funktionen
    - Seitenersetzung (bei virtuellem Speicher), dynamisches Binden
    - *copy on write* (COW), *copy on reference* (COR)

[1] „klebriges" (*sticky*) Bit: wird von Hardware gesetzt aber nicht gelöscht.

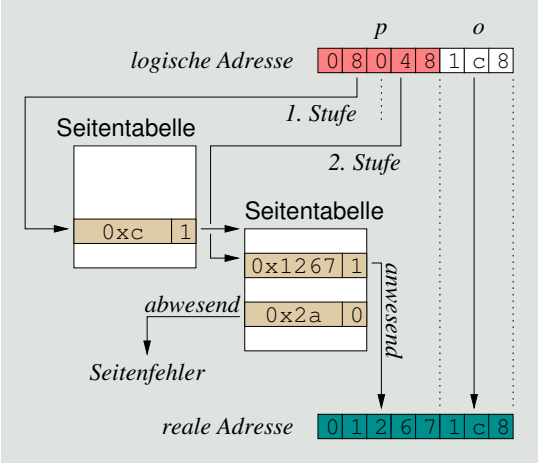
Adressabbildung I

ein- vs. mehrstufig

- angenommen, die CPU dereferenziert die Adresse 0x080481c8:
  - einstufige Abbildung



  - zweistufige Abbildung (x86)



  - *base/limit* Registerpaar (MMU) grenzt die Seitentabelle ein
  - verschieden große Seitentabellen, je nach Prozessexemplar

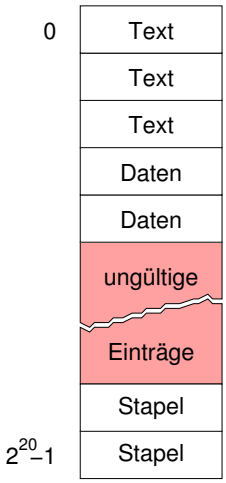
  - *base* Register (MMU) lokalisiert die Seitentabelle der 1. Stufe
  - gleich große Seitentabellen, für alle Prozessexemplare

  - ***Trap***, falls $p \geq limit$ (li.) oder ungültiger/leerer Seitendeskriptor (beide)

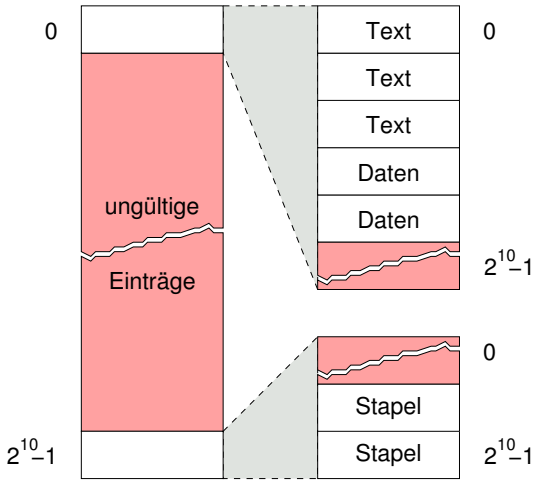
Adressabbildung ohne Tabelleneingrenzung

x86

- ein Prozess belege 12 KiB Text, 8 KiB Daten und 8 KiB Stapel
  - 4 KiB Seiten, 32-Bit Seitendeskriptor, 32-Bit Adresse/Prozessor
- einstufige Tabelle



- $2^{20} - 7$ ungültige Einträge
  - 1 Tabelle pro Prozessexemplar
  - 4 MiB Speicherplatzbedarf ☹
- zweistufige Tabelle



- $3 * 2^{10} - 9 = 3063$ ungültige Ein.
  - 3 Tabellen pro Prozessexemplar
  - 12 KiB Speicherplatzbedarf ☺

Zweidimensionaler Adressraum

segmentiert

Segmentierung *meint die Unterteilung des Adressraums in Einheiten von möglicherweise verschiedener Größe, sogenannter* ***Segmente****, die ihrerseits gleichgroße Einheiten linear aufgezählt enthalten.*

- jedes Segment S bildet eine lineare Folge fester Speichereinheiten:
  - Byte
    - S ist unbedingt zusammenhängend, auch im realen Adressraum
  - Seite
    - S ist unbedingt zusammenhängend im logischen Adressraum, aber
    - bedingt zusammenhängend im realen Adressraum
    - ↪ **seitennummerierte Segmentierung** (*paged segmentation*)
- die vom Prozess generierte Adresse *la* bildet ein Paar (S, A):
  - S ist **Segmentname** (auch Segmentnummer) ⇝ 1. Dimension
    - Wertebereich für $S = [0, 2^M - 1]$; bei IA-32: $M = 13$
  - A ist **Adresse**, auch **Versatz**, innerhalb des Segments S ⇝ 2. Dimension
    - Wertebereich für $A = [0, 2^N - 1]$
- tabellengesteuerte Abbildung von *la* mit S als **Segmentindex**
  - selektiert den für S gültigen **Segmentdeskriptor** in der **Segmenttabelle**

Segmentdeskriptor

segment descriptor

- ein von der Hardware (MMU) vorgegebener **Verbund** von Daten, der statische/dynamische **Segmenteigenschaften** beschreibt:
  - Basis
    - Segmentanfangsadresse im Haupt- bzw. Arbeitsspeicher
    - Ausrichtung (*alignment*) entsprechend der Granulatgröße
  - Limit
    - Segmentlänge als Anzahl der Granulate
    - Zahl der gültigen, linear aufgezählten Granulatadressen
  - Attribute
    - Typ (Text, Daten, Stapel)
    - Zugriffsrechte (lesen, schreiben, ausführen)
    - Expansionsrichtung (auf-/abwärts)
    - Präsenzbit
- je nach Hardware und Adressraummodell gibt es weitere Attribute
  - Privilegstufe, Klasse (*interrupt*, *trap*, *task*), Granulatgröße, ...
  - Seiten-Kachel-Tabelle (seitennummerierte Segmentierung)

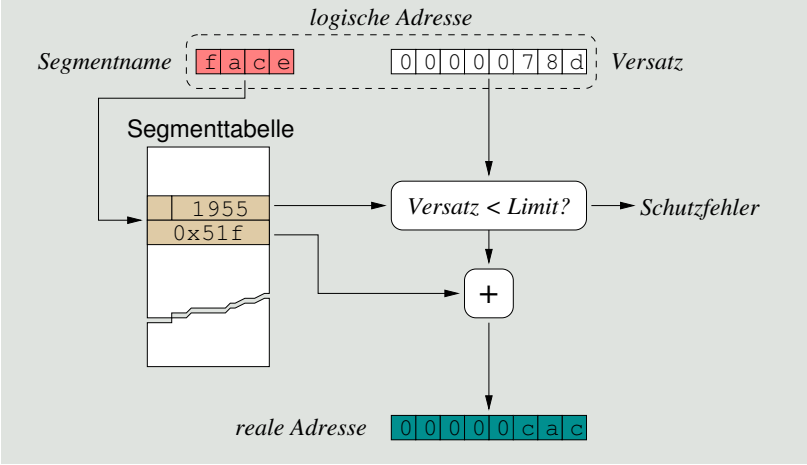
Hinweis

Ursprünglich war Segmentierung eine Technik, um mehr Hauptspeicher adressieren zu können, als es durch die Adressbreite allein möglich war. Ein prominentes Beispiel dafür war/ist der i8086: 16-Bit breite Adresse, jedoch $A_r = [0, 2^{20} - 1]$.

Adressabbildung II

byteorientiert

- angenommen, die CPU dereferenziert die Adresse `0x78d` im Segment namens `0xface` ⇝ **zweikomponentige Adresse**:



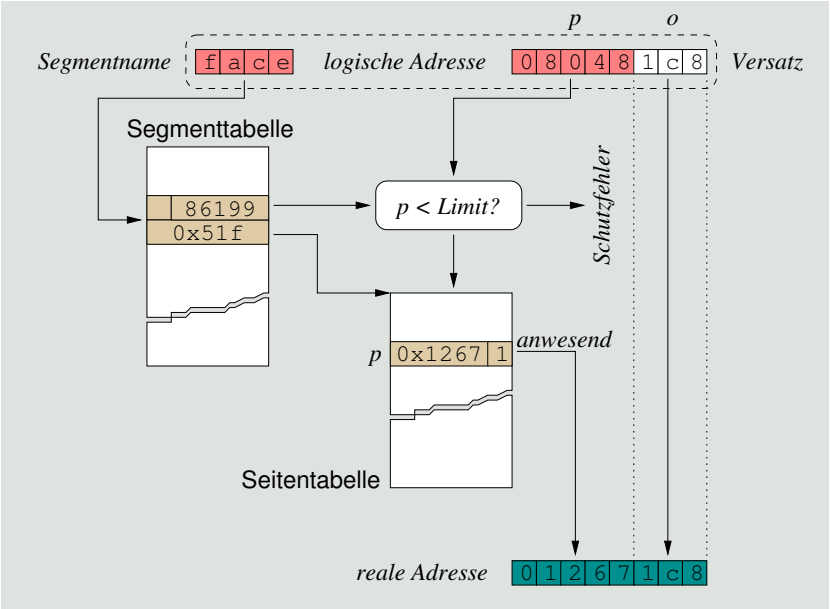
- evtl. Ausnahmen beim Abbildungsvorgang:
  - Schutzfehler (s.u.)
  - Segmentfehler wegen Abwesenheit: *swap-out*
  - Zugriffsverletzung (falsche Rechte)
- bewirken einen ***Trap***, den die MMU erzeugt

- der Tabelleneintrag $face_{16} = 64206_{10}$ liefert den Segmentdeskriptor
- die Adresse ist ein Versatz zur Segmentanfangsadresse im Hauptspeicher
  - sie ist gültig, wenn ihr Wert kleiner als die Segmentlänge ist
  - dann wird sie zur Segmentanfangsadresse addiert ⇝ **Relokation** (*relocation*)
  - ansonsten ist sie ungültig ⇝ **Schutzfehler** (*segmentation fault*)

Adressabbildung III

seitenorientiert

- **seitennummerierte Segmentierung** (*paged segmentation*):



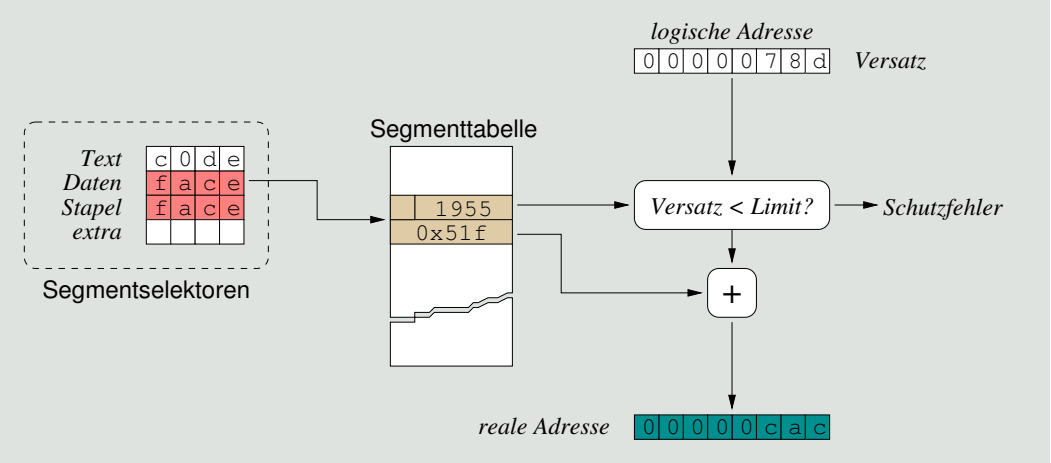
- gekachelte Segmente, grobgranular
- keine mehrstufigen Seitentabellen
- Seitentabellen verschiedener Größen
- globale oder lokale Segmenttabellen
  - pro System oder
  - pro Prozessexemplar
- nicht unkompliziert...

- ein Segment erfasst letztlich eine Seitentabelle bestimmter Größe
  - die Segmentanfangsadresse lokalisiert die Seitentabelle im Hauptspeicher
  - die Segmentlänge definiert die Größe der Seitentabelle

Adressabbildung IV

implizit segmentiert

- je nach **Zugriffsart** des in Ausführung befindlichen Befehls selektiert die MMU implizit das passende Segment:



- Befehlsabruf (*instruction fetch*) aus Textsegment
  - Operantionskode ↦ Segmentname „Text“
- Operandenabruf (*operand fetch*) aus Text-, Daten-, Stapelsegment
  - Direktwerte ↦ Segmentname „Text“
  - globale/lokale Daten ↦ Segmentname „Daten“ ≡ „Stapel“

↪ Programme können weiterhin **einkomponentige Adressen** verwenden

Übersetzungspuffer

translation lookaside buffer (TLB)

*Jede Dereferenzierung bedeutet den indirekten Zugriff über eine oder mehrere Tabellen im Hauptspeicher, der impraktikabel ist, wenn keine Vorkehrung zur **Latenzverbergung** getroffen wird.*

- **Zwischenspeicher** (*cache*) für das Übersetzungsergebnis, d.h., einer Unter- oder Obermengen von Deskriptoren[2]
  - Assoziativspeicher für eine kleine Anzahl (8 – 128) von Puffereinträgen
  - Segment- bzw. Seitenindex (der virtuellen Adresse) als Suchschlüssel
- ein **Umsetzungsfehler** (*lookup miss*) führt zur **Tabellenwanderung** (*table walk*), die hard- oder softwaregeführt geschieht

  hardwaregeführt
  - die CPU läuft die Tabellen ab
  - mittelbarer Trap, bei erfolgloser Tabellenwanderung
  - ↪ in Hardware implementierte MMU (x86, PPC)

  softwaregeführt
  - unmittelbarer Trap, beim Umsetzungsfehler
  - das Betriebssystem läuft die Tabellen ab
  - ↪ in Software implementierte MMU (MIPS, Alpha)
- letztere hat eine höhere **Auffüllzeit**, aber auch höhere **Flexibilität** [11]

[2] Erstmalig umgesetzt in IBM System/370 [3, 1].

Abstraktion von der Speicherlokalität

*Eine virtuelle Adresse erbt alle Eigenschaften einer logischen Adresse und erlaubt darüber hinaus **ortstransparente Zugriffe** auf externen Speicher — desselben oder eines anderen Rechensystems.*

- **lose Bindung** zwischen Adresse und durch sie adressierten Entität:

  logische Adresse
  - entkoppelt von der Lokalität im **Hauptspeicher**
  - ermöglicht **dynamisches Binden** aktiver Prozesse
  - erlaubt **Tauschen** (*swapping*) inaktiver Prozesse

  virtuelle Adresse
  - ist eine logische Adresse und geht darüber hinaus, sie:
  - entkoppelt von der Lokalität im **Arbeitsspeicher**
  - erlaubt **Seitenumlagerung** (*paging*) aktiver Prozesse
- die Adressabbildung impliziert **partielle Interpretation** der Zugriffe
  - steuerndes Mittel ist das **Präsenzbit** eines Segments/einer Seite

    0 – abwesend, verursacht einen **Zugriffsfehler** (*access fault*) ⇝ ***Trap***

    1 – anwesend, unterbricht die Dereferenzierung nicht
- **Ausnahmebehandlung** und Wiederaufnahme des Prozesses
  - das Betriebssystem sorgt für die Anwesenheit des Segments/der Seite und
  - die CPU wird instruiert, den unterbrochenen Befehl zu wiederholen

Gliederung

Einführung
- Rekapitulation

Adressräume
- Real
- Logisch
- Virtuell

Mehradressraumsysteme
- Virtualität
- Exklusion
- Inklusion

Zusammenfassung

Adressvirtualisierung

partielle Virtualisierung

Definition (Virtualität [12])

Die Eigenschaft einer Sache, nicht in der Form zu existieren, in der sie zu existieren scheint, aber in ihrem Wesen oder ihrer Wirkung einer in dieser Form existierenden Sache zu gleichen.

- **Virtualisierung des <u>realen</u> Adressbereichs** – nicht Hauptspeichers!
  - i Vervielfachung von $A = [0, 2^N - 1]$
    - komplett A für Betriebssystem und allen Maschinenprogrammen, jeweils
  - ii Einrichtung von $A_t = [0, 2^N - 1]$, Vervielfachung von $A_p \subset A_t$
    - komplett A_t (total) für das Betriebssystem
    - komplett A_p (partiell) für alle Maschinenprogramme, jeweils
- Adressen dieser Bereiche sind nicht wirklich (physisch), wohl aber in ihrer Funktionalität vorhanden
  - hinter jeder dieser Adresse steht eine speicherabbildbare Entität
  - sie referenzieren Entitäten der Programmtexte (d.h., Befehle) oder -daten

Private Adressräume

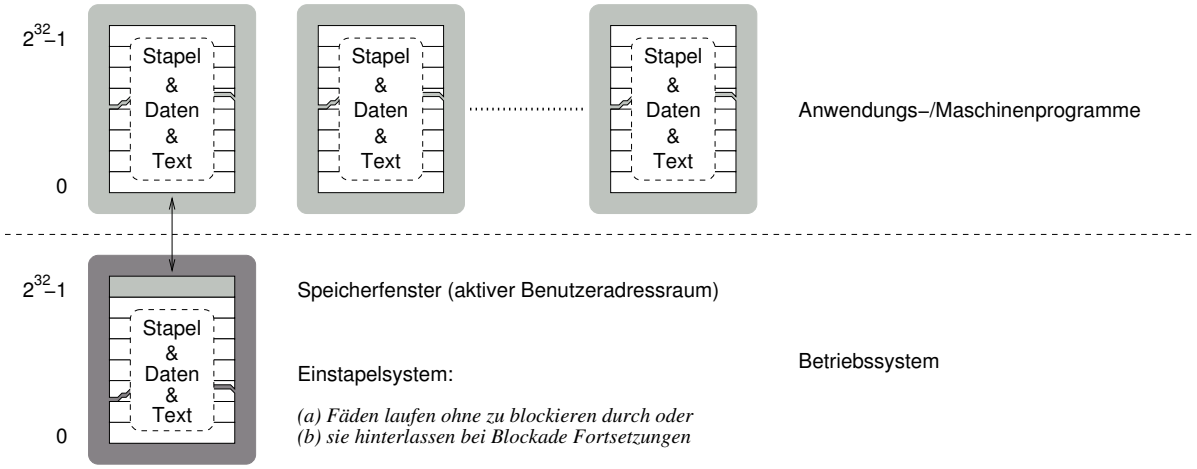
Illusion** von einem eigenen physischen Adressraum für Betriebssystem und Maschinenprogramme* $\rightsquigarrow$ ***Exklusion

- **Vervielfachung** des Adressbereichs $A = [0, 2^N - 1]$
  - wobei N bestimmt ist durch die reale Adressbreite des Prozessors
  - evolutionär betrachtet galt/gilt z.B. für $N = 16, 20, 24, 31, 32, 48, 64$
- **Spezialhardware**[3] verhindert ein Ausbrechen von Prozessen aus A
  - dies gilt für alle durch das Betriebssystem verwalteten Prozesse, *also*
  - sowohl für Maschinenprogramme als auch für das Betriebssystem selbst
- evtl. Datenaustausch zwischen Programmen erfordert **Spezialbefehle**
  - des Betriebssystems für die Maschinenprogramme *und* $\mapsto$ Ebene $_3$
    - Systemaufrufe zur Interprozesskommunikation oder Adressbereichsabbildung
  - der CPU für die Betriebssystemprogramme $\mapsto$ Ebene $_2$
    - privilegierte Befehle zum Lese-/Schreibzugriff auf den Benutzeradressraum
- im Vordergrund steht die **strikte Isolation** von ganzen Adressräumen

[3] MMU, aber ebenso eine MPU (*memory protection unit*).

Programmierte Mitbenutzung

partiell



- zeitgenössisch **fensterbasierter Ansatz** zum Datenaustausch
  - horizontal
    - Interprozesskommunikation (Nachrichtenversenden)
    - seitenbasierte Mitbenutzung
  - vertikal
    - Zugriffe auf den Benutzeradressraum durch **Speicherfenster**
- prominentes Beispiel eines Betriebssystems der Art:
  - OS X
    - Hybridkernansatz, ereignis-/prozedurbasiert, 512 MiB Fenster

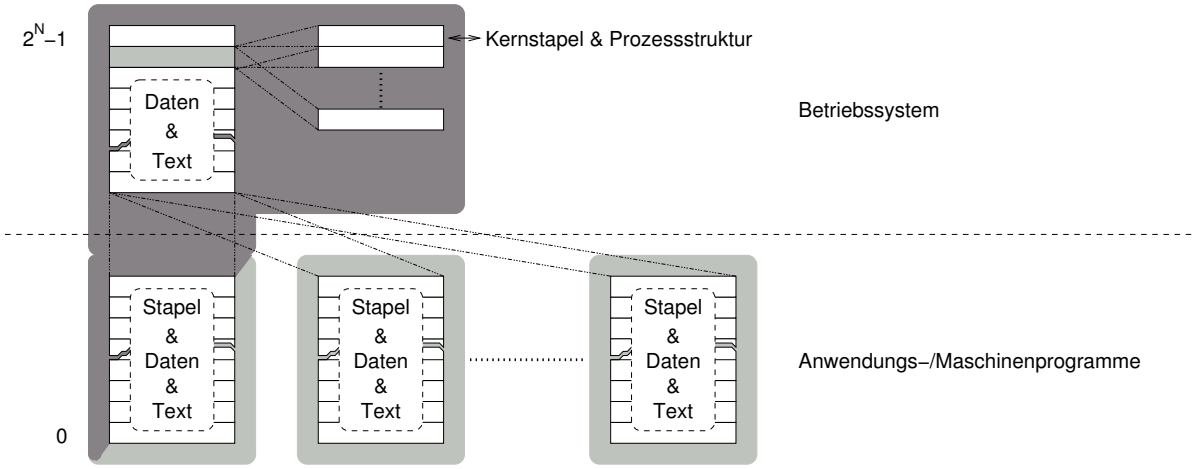
Partiell private Adressräume

***Illusion** von einem eigenen physischen Adressraum bzw. -bereich für die Maschinenprogramme* $\rightsquigarrow$ ***Inklusion** des Betriebssystem(kern)s*

- **Vervielfachung** des Adressbereichs $A_p \subset A_t$
  - A_t ist der dem Betriebssystem *total* zugeordnete Adressbereich
    - existiert einfach, aber mit A_p als integrierten variablen (mehrfachen) Anteil
  - A_p ist der einem Maschinenprogramm in A_t *partiell* zugeordnete Bereich
    - existiert mehrfach, einmal für jedes Anwendungs- bzw. Maschinenprogramm
- der Benutzeradressraum ist ein Teil (genauer: eine echte Teilmenge) des Betriebssystemadressraums
  - die MMU verhindert ein Ausbrechen von Prozessen aus A_p und A_t, nicht jedoch deren Eindringen heraus aus $A_t - A_p$ und hinein in A_p
    - bedingter Schreibschutz von A_p für A_t dämmt Betriebssystemfehler ein
  - dabei erstreckt sich A_p über den oberen oder unteren Bereich von A_t
  - ein Prozesswechsel zwischen A_p bedingt das Umschalten der MMU
- im Vordergrund steht, **weniger Adressraumwechsel** hervorzurufen

Implizite Mitbenutzung

gekachelt



- zeitgenössisch **partitionsbasierter Ansatz** zum Datenaustausch
  - horizontal: Interprozesskommunikation, Segment-/Seitenmitbenutzung
  - vertikal: **speicherabgebildeter Zugriff** auf den Benutzeradressraum
- prominente Beispiele von Betriebssystemen der Art:
  - Linux: monolithisch, prozedurbasiert
  - OS X, NT: Hybridkernansatz, prozess-/prozedurbasiert

Partitionierung des Adressbereichs

Inklusion des Benutzeradressraums in den Betriebssystemadressraum ist nur bei hinreichend großem N ein sinnvoller Ansatz

- das Modell wurde attraktiv mit Adressbreiten von $N \geq 30$ Bits
  - also für reale Adressbereiche ab 1 GiB Speicherumfang
- typische Aufteilung von $A = [0, 2^{32} - 1] = 4$ GiB:
  - gleich
    - 2 GiB jeweils für Benutzer- und Betriebssystemadressraum
    - NT
  - ungleich
    - 3 GiB Benutzer- und 1 GiB Betriebssystemadressraum
    - Linux, NT (Enterprise Edition)
  - steht und fällt mit der Größe von Benutzerprogrammen/-prozessen

Inklusion bedeutet aber eben auch, dass die Benutzerprozesse dem Betriebssystem ein ***stärkeres Vertrauen*** *schenken müssen*

- Schreibschutz auf A_p legen und nur bei Bedarf zurücknehmen/lockern
  - sonst sind Zeigerfehler in $A_t - A_p$ verheerend für Programme in A_p
  - aber auch implizit erlaubte Lesezugriffe verletzten die **Privatsphäre**...

Gliederung

Einführung
- Rekapitulation

Adressräume
- Real
- Logisch
- Virtuell

Mehradressraumsysteme
- Virtualität
- Exklusion
- Inklusion

Zusammenfassung

Resümee

- **Prozessadressräume** sind (a) real, (b) logisch oder (c) virtuell
  (a) lückenhafter, wirklicher Hauptspeicher
  (b) lückenloser, wirklicher Hauptspeicher
  (c) lückenloser, scheinbarer Hauptspeicher
  - Arbeitsspeicher liegt im Vordergrund (a, b) bzw. Hintergrund (c)
- logische/virtuelle Adressräume sind **seiten- oder segmentorientiert**
  - d.h., sie sind eine Aufzählung ein- oder zweidimensionaler Adressen
    eindimensional – Tupel (Seitennummer, Versatz)
    zweidimensional – Paar (Segmentnummer, Adresse) bzw.
    – Paar (Segmentnummer, (Seitennummer, Versatz))
  - letztere Paarung gilt für die **seitennummerierte Segmentierung**
- **Mehradressraumsysteme** vervielfachen den realen Adressbereich
  - implementieren (total/partiell) private Adressräume
  - Informationsaustausch zwischen Betriebssystem- & Benutzeradressraum:
    - fensterbasiert, bedarfsorientierte Einblendung von Adressraumabschnitten
    - spezialbefehlbasiert, selektives Kopieren von Maschinenwörtern
    - adressraumgeteilt, direkter Zugriff auf kompletten Benutzeradressraum

Literaturverzeichnis I

[1] Case, R. P. ; Padegs, A. :
Architecture of the IBM System/370.
In: *Communications of the ACM* 21 (1978), Jan., Nr. 1, S. 73–96

[2] Hildebrand, D. :
An Architectural Overview of QNX.
In: *Proceedings of the USENIX Workshop on Micro-kernels and Other Kernel Architectures (USENIX Microkernels)* USENIX Association, 1992. – ISBN 1–880446–42–1, S. 113–126

[3] IBM Corporation (Hrsg.):
IBM System/370 Principles of Operation.
Fourth.
Poughkeepsie, New York, USA: IBM Corporation, Sept. 1 1974.
(GA22-7000-4, File No. S/370-01)

[4] Kleinöder, J. ; Schröder-Preikschat, W. :
Prozesse.
In: [6], Kapitel 6.1

[5] Kleinöder, J. ; Schröder-Preikschat, W. :
Speicher.
In: [6], Kapitel 6.2

Literaturverzeichnis II

[6] Kleinöder, J. ; Schröder-Preikschat, W. ; Lehrstuhl Informatik 4 (Hrsg.):
Systemprogrammierung.
FAU Erlangen-Nürnberg, 2015 (Vorlesungsfolien)

[7] Lions, J. :
A Commentary on the Sixth Edition UNIX Operating System.
The University of New South Wales, Department of Computer Science, Australia :
`http://www.lemis.com/grog/Documentation/Lions`, 1977

[8] Lions, J. :
UNIX Operating System Source Code, Level Six.
The University of New South Wales, Department of Computer Science, Australia :
`http://v6.cuzuco.com`, Jun. 1977

[9] Quantum Software Systems Ltd. (Hrsg.):
QNX Operating System User's Manual.
Version 2.0.
Toronto, Canada: Quantum Software Systems Ltd., 1984

[10] Schröder, W. :
Eine Familie von UNIX-ähnlichen Betriebssystemen – Anwendung von Prozessen und des Nachrichtenübermittlungskonzeptes bem strukturierten Betriebssystementwurf, Technische Universität Berlin, Diss., Dez. 1986

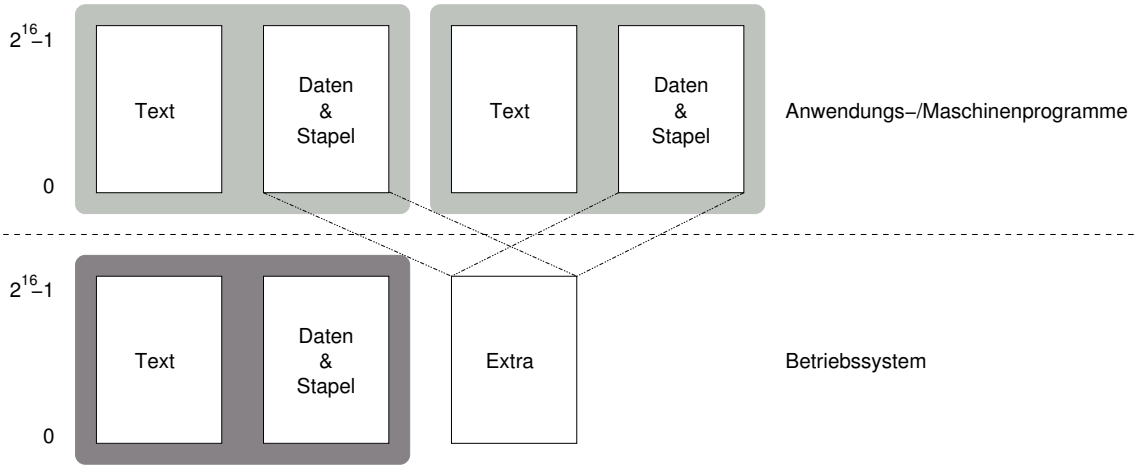
Literaturverzeichnis III

[11] Uhlig, R. ; Nagle, D. ; Stanley, T. ; Mudge, T. ; Sechrest, S. ; Brown, R. :
Design Tradeoffs for Software-Managed TLBs.
In: *ACM Transactions on Computer Systems* 12 (1994), Aug., Nr. 3, S. 175–205

[12] Wikipedia:
Virtualität.
`https://de.wikipedia.org/wiki/Virtualit{ä}t`, Aug. 2015

Programmierte Mitbenutzung

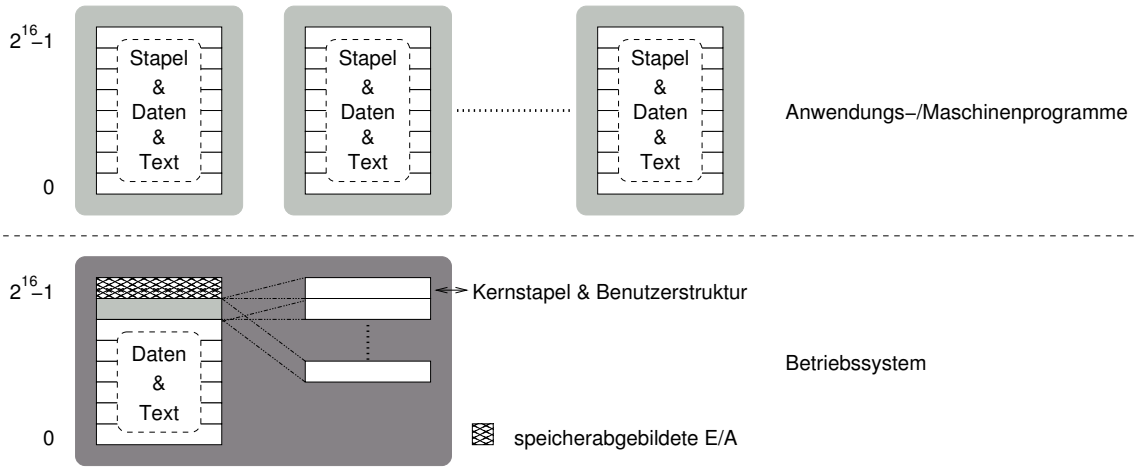
total



- antiquiert (i8086) **fensterbasierter Ansatz** zum Datenaustausch
  - horizontal ▪ Interprozesskommunikation (Nachrichtenversenden)
  - vertikal ▪ Zugriffe auf den Benutzeradressraum mittels **Extrasegment**
- Beispiele von (mikrokernbasierten) Betriebssystemen der Art:
  - QNX [9] ▪ ereignisbasiert, vgl. auch [2]
  - AX [10] ▪ ereignis-/prozedurbasiert, QNX-kompatibel

Programmierte Fernzugriffe

selektiv



- antiquiert (PDP 11/40) **kopiebasierter Ansatz** zum Datenaustausch
  - horizontal ▪ Interprozesskommunikation (Nachrichtenversenden, *pipe*)
    - ▪ seitenbasierte Mitbenutzung in Inkrementen von 64 Bytes
  - vertikal ▪ Zugriffe auf den Benutzeradressraum mittels **Spezialbefehle**
    - ▪ *move from/to previous instruction space* (`mfpi`/`mtpi`)
- prominentes Beispiel eines (monolithischen) Betriebssystems der Art:
  - UNIX ▪ Version 6 [8, 7], prozedurbasiert