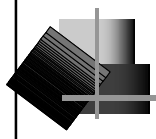
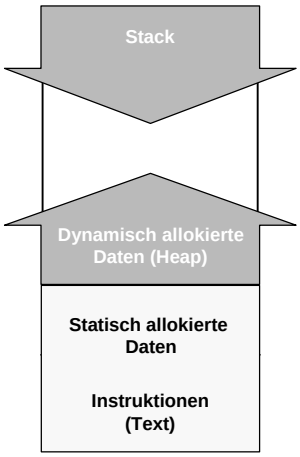


# 1. Speicher

1

## Typische Nutzung eines Adreßraums

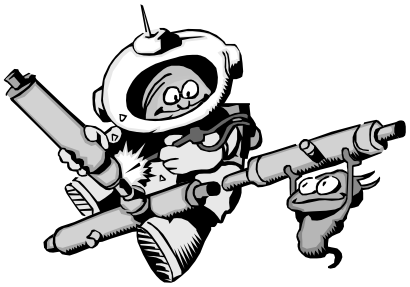
- Textbereich relativ klein
- Sehr großer Abstand zwischen Heap und Stack
- Keine “Verunreinigungen” durch:
  - E/A-Bereiche
  - nicht bestückte Adreßbereiche
  - fremde Kontrollflüsse
- Idealvorstellung:
  - maximale Größe (z.B. 4 GB)
  - vollständig adressierbar (ungeachtet des tatsächlichen Speicherausbaus)
  - exklusive Nutzung durch einen Kontrollfluß
  - Isolation von anderen Adreßräumen




2

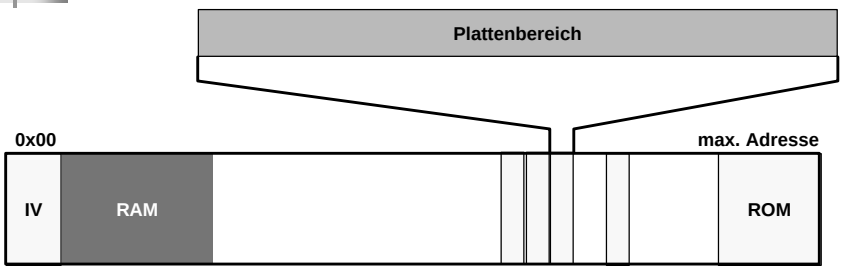
## Experiment 2

- Wie sieht der Adreßraum in Linux und Windows 2000 aus?



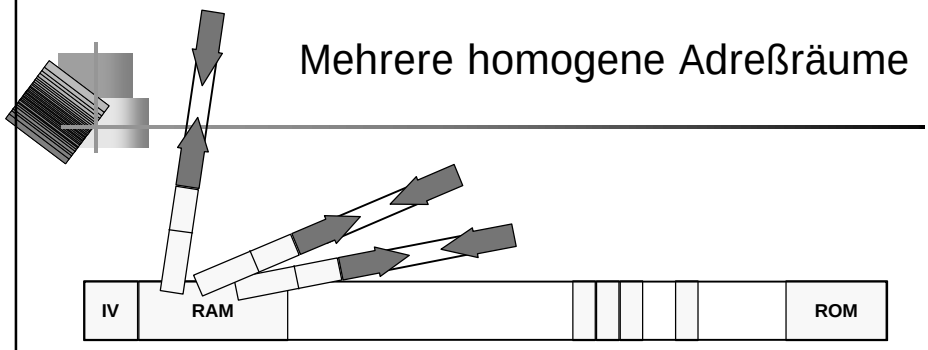
3

## Adreßräume ...



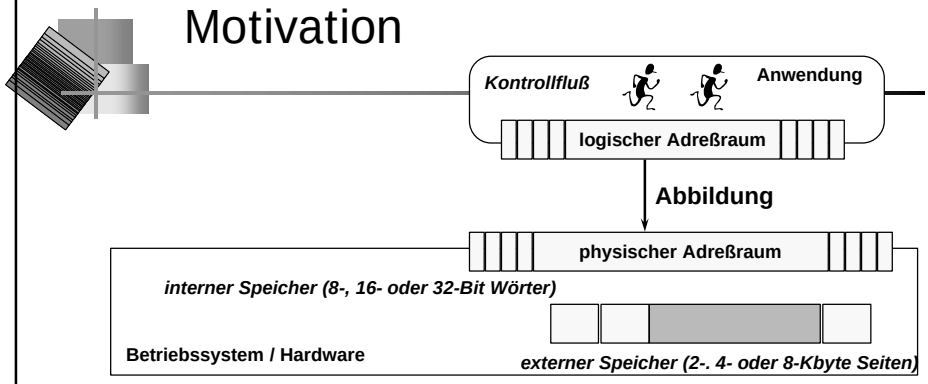

- Realer Adreßraum enthält viele große Lücken
  - Interruptvektoren, Sprungtabellen, Startadresse nach Reset
  - RAM-Speicher
  - Boot-ROM
  - Bereiche für speicherbasierte E/A-Geräte

4



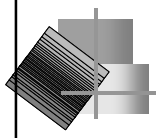

- Jeder Adreßraum selbst so groß wie der real adressierbare Adreßraum (z.B. 32 Bit)
- Lösungsansätze?
- Tips:
  - typischerweise nur Anfang und Ende benutzt
  - Lokalitätsprinzip

5




- Reale Adressierung
  logischer Adreßraum gleich physischer Adreßraum
- Logische (Virtuelle) Adressierung
  logischer Adreßraum ungleich physischer Adreßraum

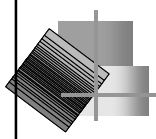
6



# Ziele

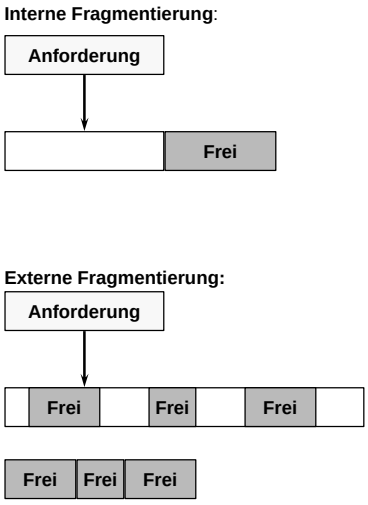
- Unterstützung mehrerer Adreßräume
  - Mehrerer Kontrollflüsse (= Bereitstellung der Stackbereiche)
- Schutz innerhalb eines Adreßraums
- Schutz (Protection)
  - Betriebssystem vor fehlerhaften Anwendungen geschützt
  - Schutz zwischen verschiedenen Anwendungen
- Mehr Adreßraum als real vorhanden
  - Einbeziehen von externem Speicher
  - Überlagern und Aus/Einlagern von ganzen Adreßräumen oder Teilen davon (Seiten)
- Fortgeschrittene speicherbasierte E/A-Techniken

7

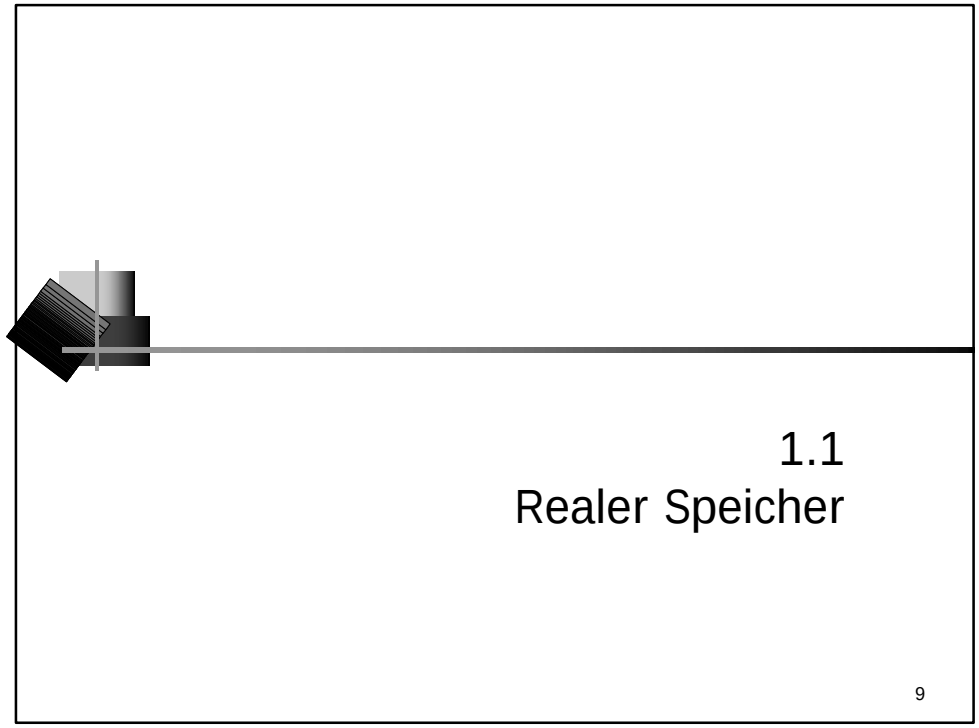


# Fragmentierung

- Speicherverschnitt bei der Vergabe eines neuen Speicher-bereichs (Adreßraum)
- Interne Fragmentierung:
  - Zuteilungseinheit größer als Anforderung
- Externe Fragmentierung:
  - Verfügbare Reservoir an zuteilbaren Bereichen wird zerstückelt
  - Anforderung trotz genügend freiem Speicher nicht erfüllbar

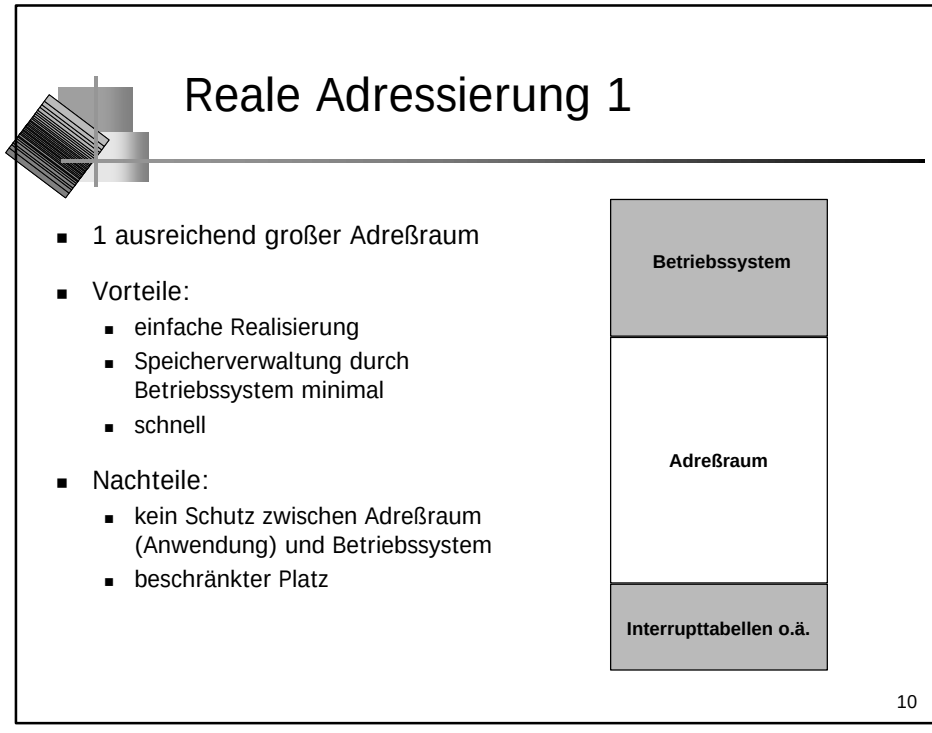



8



# 1.1
# Realer Speicher

9



## Reale Adressierung 1

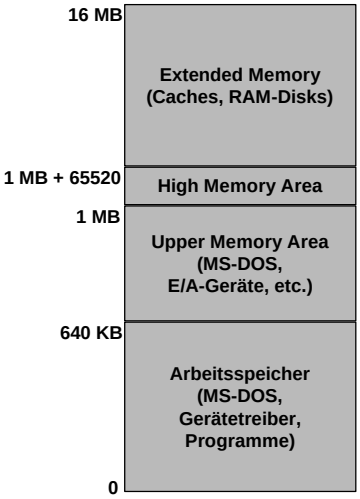
- 1 ausreichend großer Adreßraum
- Vorteile:
  - einfache Realisierung
  - Speicherverwaltung durch Betriebssystem minimal
  - schnell
- Nachteile:
  - kein Schutz zwischen Adreßraum (Anwendung) und Betriebssystem
  - beschränkter Platz




10

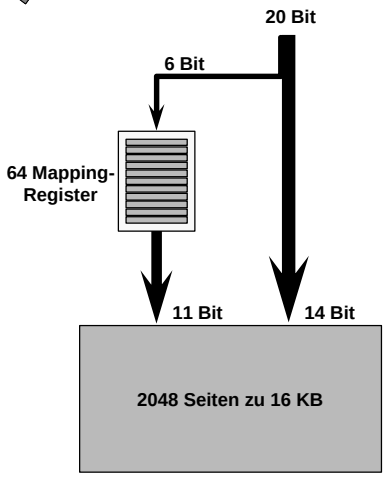
# Beispiel: MS-DOS

- Segmentbasierte Adressierung
  - Datenstrukturen > 64 KB teuer
  - Fernaufrufe (Prozeduren außerhalb des Segments ebenfalls teuer
- UMA war von IBM bei PC's für E/A-Anbindung gedacht
  - Welchem Programm könnten jemals die unteren 640 KB nicht reichen?
- Durch Adressierungstricks kann MS-DOS in den HMA-Bereich
  - bei 8088 nicht vorgesehen, d.h. MS-DOS in diesem Bereich nicht kompatibel
- Extended Memory nur bedingt nutzbar (vgl. Overlays)




11

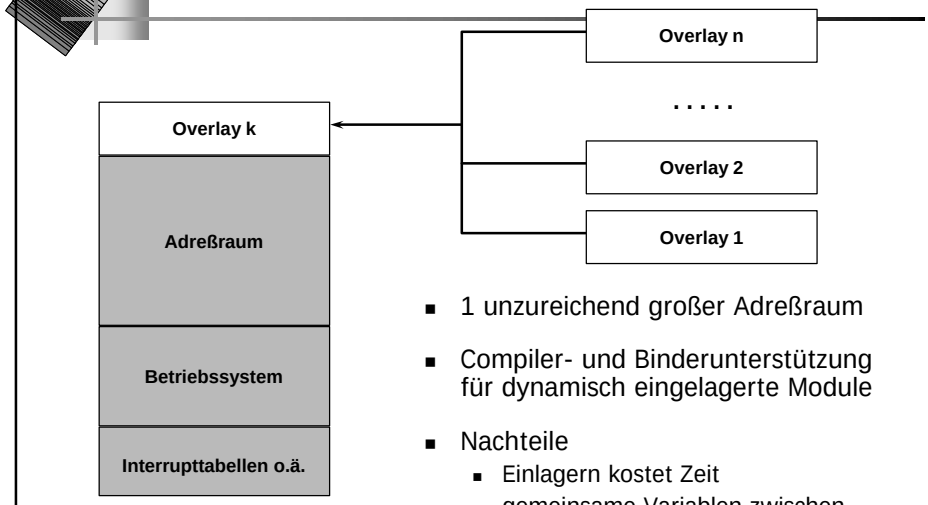
# MS-DOS: Expanded Memory (EMS)




- Verfügbares Speicher-Layout zu beschränkt
- Bank Switching:
  - 1 MB PC-Speicher unterteilt in 64 Seiten zu je 16 KB
  - max. 32 MB EMS unterteilt in 2048 Seiten zu je 16 KB
  - Mapping-Register blenden EMS-Seiten in den Adreßraum
- Spezielle Hardware notwendig
- Früher gängige Technik
  - z.B. Apple II

12

## Reale Adressierung 2




- 1 unzureichend großer Adreßraum
- Compiler- und Binderunterstützung für dynamisch eingelagerte Module
- Nachteile
  - Einlagern kostet Zeit
  - gemeinsame Variablen zwischen verschiedenen Overlays?

13

## Overlay-Technik

- Globale Variable:

```
int overlay = 0;
```

- Aufruf einer Funktion `f` in einem Overlay `ziel_ov`:

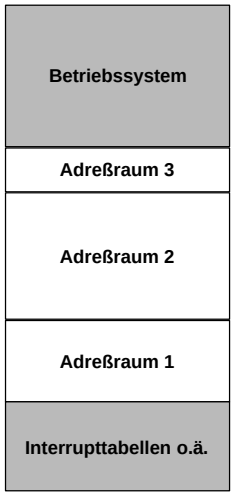
```
Parameter auf den Keller;

if (overlay != ziel_ov) {
    Lade Ziel-Overlay;
    overlay = ziel_ov;
    }

call f;
```

- Verschiedene Realisierungen
  - Funktion zu Overlay-Zuordnung meist durch Programmierer
  - Analyse des Aufrufgraphen möglich

14

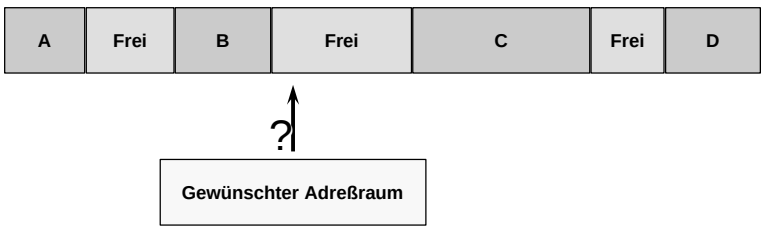
# Reale Adressierung 3

- mehrere Adreßräume
- verschiedene reale Adreßbereiche für Anwendungen
  - bereits beim Binden
  - Relokation beim Laden
- Nachteile
  - Kein Schutz zwischen Anwendungen
  - maximale dynamische Speicheranforderungen a priori bekannt
  - Fragmentierung




15

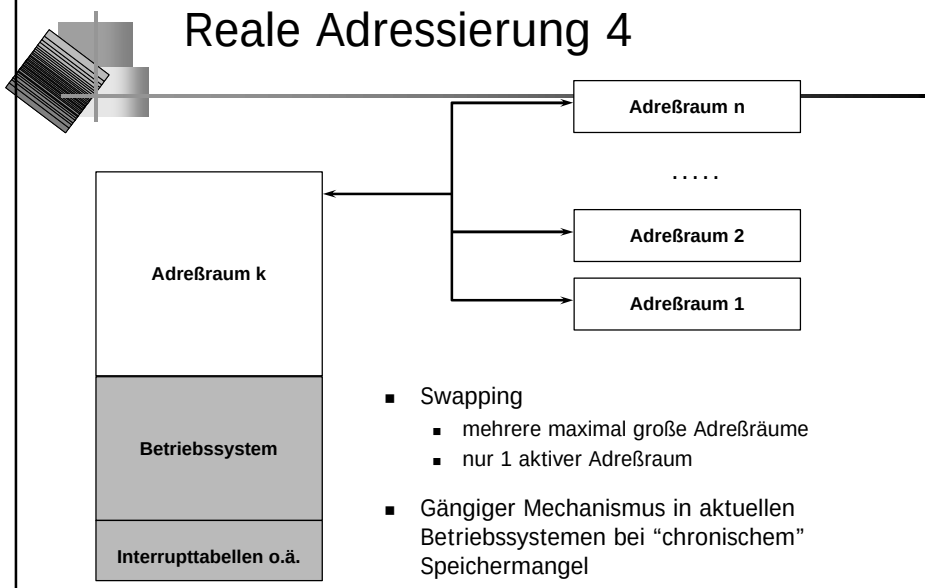
# Externe Fragmentierung




- Freier Bereich ausreichend groß
- Kann verschoben werden?
  - z.B. B nach links und C nach rechts?

16

## Reale Adressierung 4



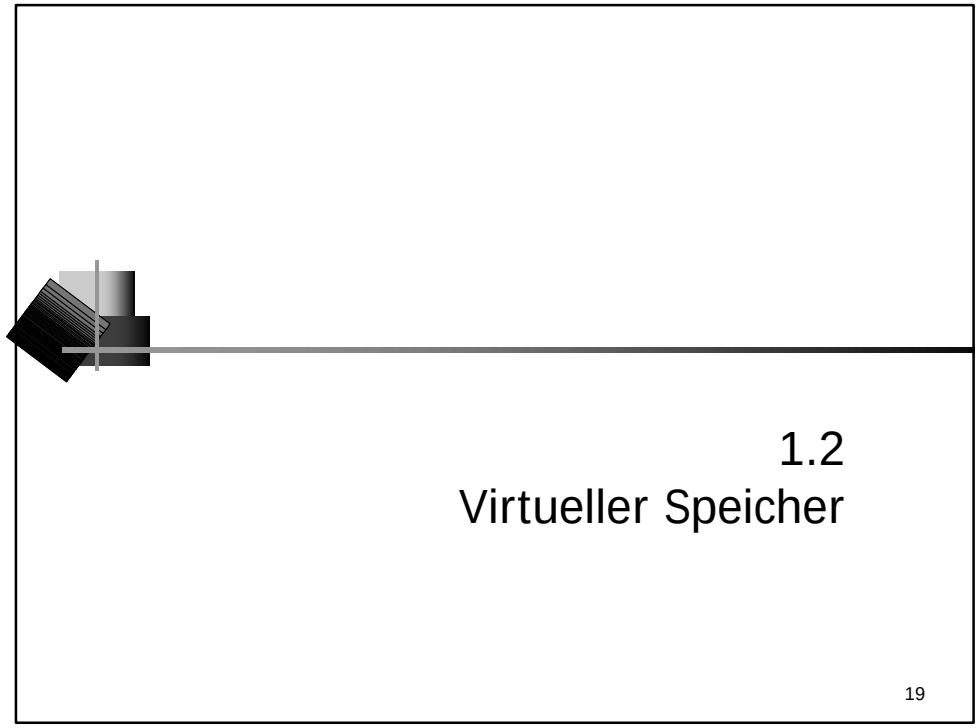

- Swapping
  - mehrere maximal große Adreßräume
  - nur 1 aktiver Adreßraum
- Gängiger Mechanismus in aktuellen Betriebssystemen bei “chronischem” Speichermangel
- Ein/Auslagern zeitaufwendig

17

## Swapping

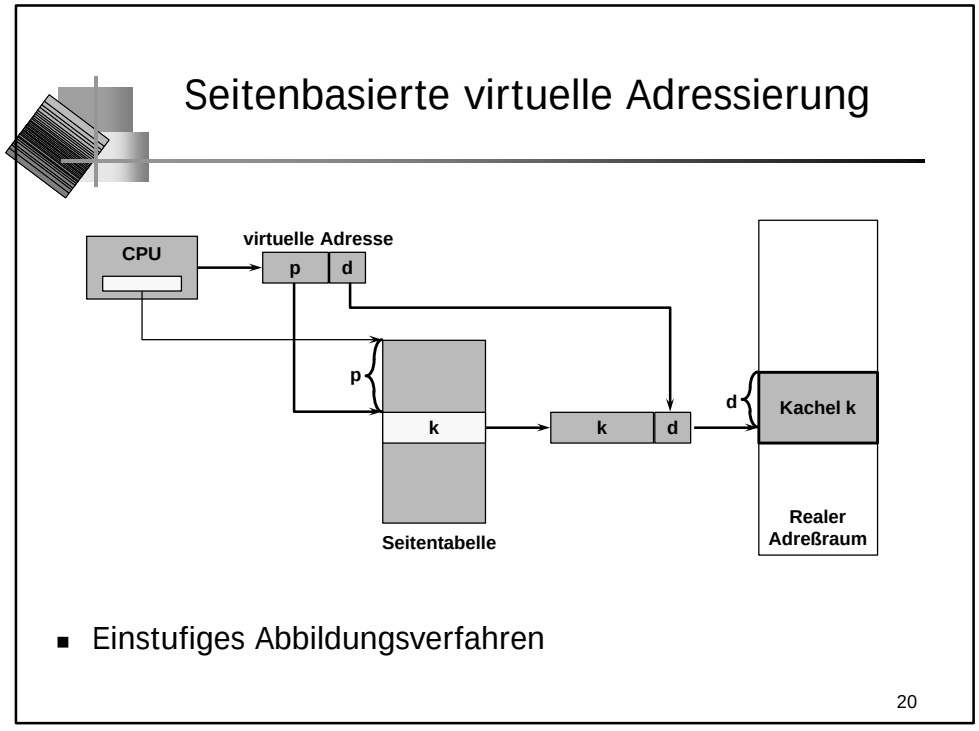
- Unterschiede zur Overlay-Technik
  - Hohe Granularität: ganze Adreßräume statt einzelne Module
  - Änderungen beim Adreßraumwechsel wieder zurückschreiben
- Hilfreiche Anforderungen an Hardware
  - Welche Dinge wurden verändert?

18



# 1.2
# Virtueller Speicher

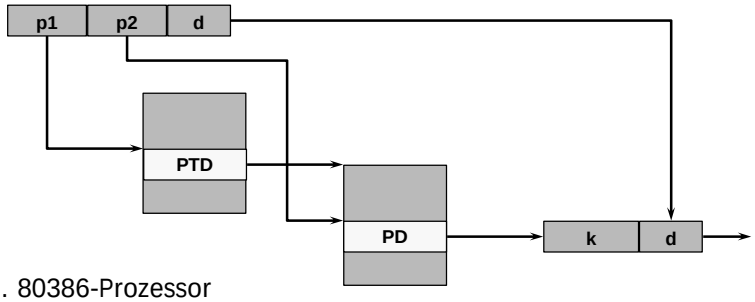
19




- Einstufiges Abbildungsverfahren

20

## Zwei- und mehrstufige Verfahren

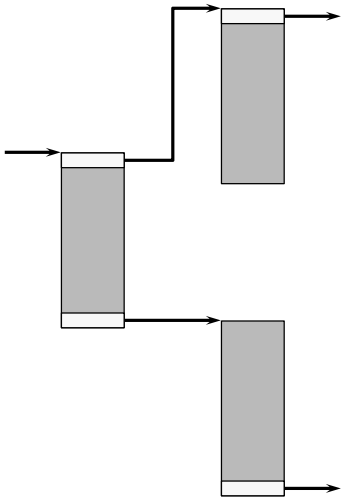



- z.B. 80386-Prozessor
  (p1 = p2 = 10 Bit Länge)
- PTD = Seitentabellen-Deskriptor
  - zeigt auf die nächst-tiefere Seitentabelle
- PD = Seiten-Deskriptor
  - zeit auf eine Kachel (falls aktuell eingeblendet)

21

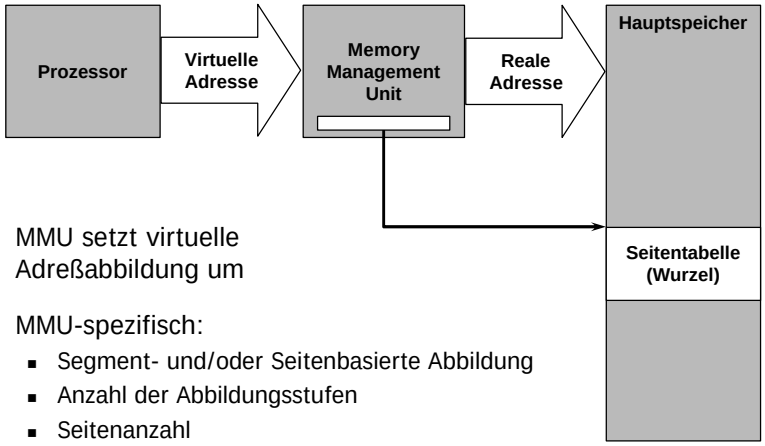
## Vorteil mehrstufiger Verfahren

- Nur die Wurzeltabelle muß initial angelegt sein
- Seitentabellen der tieferen Stufen nur bei Bedarf
  - z.B. Erstes und letztes Byte benutzt (p1=p2=10):
    $3 * 2^{10} * 6$ Byte = 18 Kbyte



22

## Die MMU




- MMU setzt virtuelle Adreßabbildung um
- MMU-spezifisch:
  - Segment- und/oder Seitenbasierte Abbildung
  - Anzahl der Abbildungsstufen
  - Seitenanzahl

23

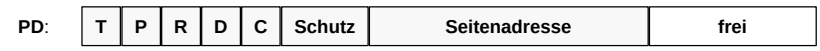
## Seitentabellen-Deskriptoren

| PTD: | T | P | Basis der Seitentabelle | frei |
|---|---|---|---|---|

- T-Bit ermöglicht Unterscheidung:
  - Seitentabellen-Deskriptor (PTD)
  - Seiten-Deskriptor (PD)
- P-Bit:
  - Seiten- oder Segmenttabelle im Hauptspeicher vorhanden (P=1)

24

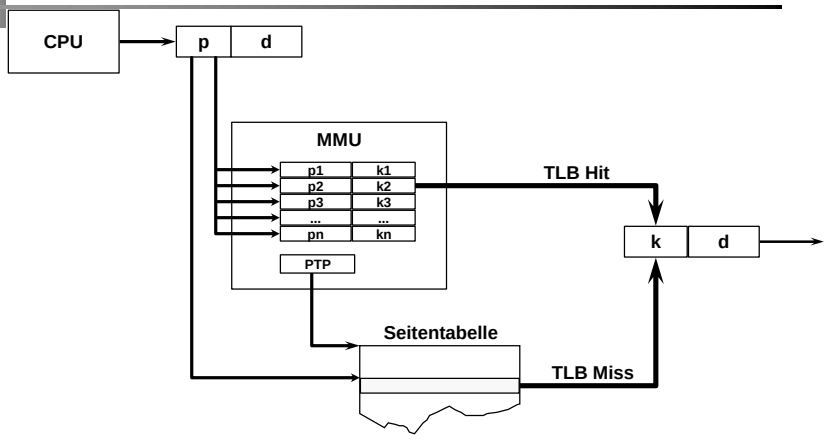
# Seiten-Deskriptoren




- Schutzbits:
  - Legen die erlaubten Zugriffsmodi fest (Lesen, Schreiben, Ausführen, ...)
  - Alle Bits 0 = keine Zugriffsform erlaubt
  - Schutzverletzung beim Zugriff bedeutet Fehler
- C-Bit (Cache Disable Bit)
  - Inhalte der Seite dürfen in Caches nicht zwischengespeichert werden
- D-Bit (Dirty-Bit)
  - Seit letztem Zurücksetzen fand ein Schreibzugriff statt
- R-Bit (Referenced-Bit)
  - Seit letztem Zurücksetzen wurde auf Seite in irgendeiner Form zugegriffen

25

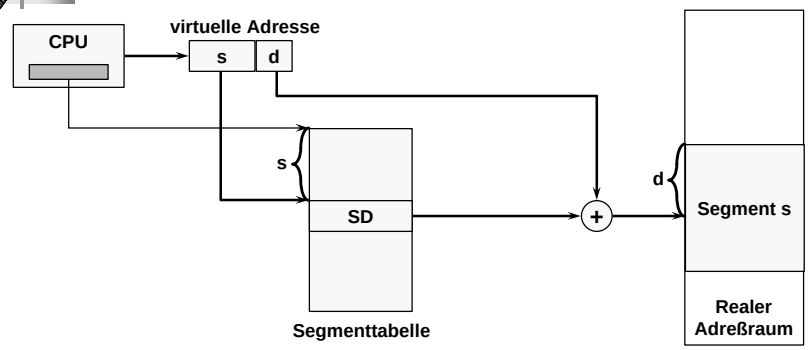
# Der TLB (Translation Lookaside Buffer)




- TLB = Assoziative Suche nach Einträgen für virtuelle Seiten
- Voraussetzung für hohe Trefferquote: Lokalität

26

## Segmentbasierte virtuelle Adressierung




- Segment-Deskriptor (SD):
  - Basisadresse (Segmentanfang)
  - Länge
- Segment zusammenhängend im realen Adreßraum

27

## Vergleich

Seitenbasierte Verfahren

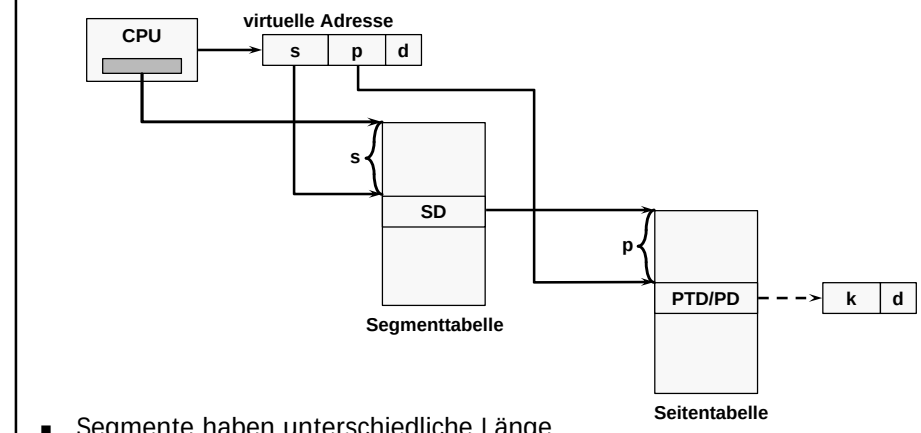
- Seiten fester Länge
  - Viele Seiten
  - Tabelle proportional zu Seitenbedarf
- Transparent für Anwendungen
- Fragmentierung:
  - Externe: Nein
  - Interne: Ja
- Adreßrechnung:
  - Abbildung

Segmentbasierte Verfahren

- Segmente variabler Länge
  - Wenig Segmente
  - Tabelle proportional zu Segmentbedarf
- Segmentstruktur für Anwendungen sichtbar
- Fragmentierung:
  - Externe: Ja
  - Interne: Nein
- Adreßrechnung:
  - Abbildung
  - Addition

28

## Segment- und Seitenbasierte Verfahren




- Segmente haben unterschiedliche Länge
- Externe und interne Fragmentierung beschränkt auf 1 Seite

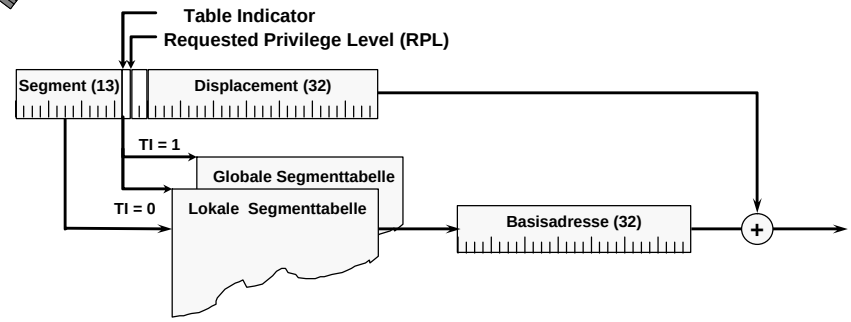
29

## Die 80386 MMU

- On-Chip MMU
- Unterstützung für Segmente und Seiten (einzeln abschaltbar)
  - Real: 4 Gbyte linearer Adreßraum
  - Segmente: 64 Tbyte Adreßraum pro Prozeß
    - 2*8192 Segmente
    - max. 4 Gbyte pro Segment
  - Seiten: 4 Gbyte zweistufiger, seitenbasierter Adreßraum
    - $2^{10}$ Einträge jeweils in der ersten und zweiten Stufe
    - 4 Kbyte große Seitentabellen
    - 4 Kbyte Seitengröße
  - Segmente und Seiten: 64 Tbyte Adreßraum
    - 2*8192 Segmente
    - für jedes Segment zweistufige, seitenbasierte Adreßabbildung

30

# 80386: Segmentierung




- $2^{46}$ Bytes (=64 Tbyte) virtueller Adreßraum pro Prozess:
  - 32 Tbyte globales Segment (für alle Prozesse gleich, TI=1)
  - 32 Tbyte lokales Segment

31

# 80386: Segment-Deskriptor

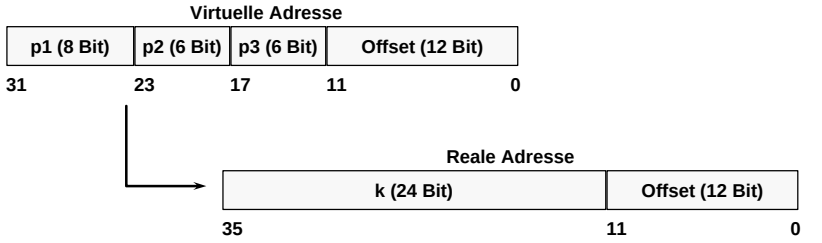
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Segmentbasis 31..24 | G | 000 | Länge 19..16 | P | DPL | 1 | Typ | A | Segmentbasis 23..16 |
| Segmentbasis 15..0 | | | | Segmentlänge 15..0 | | | | | |

- Maximale Segmentlänge: 20 Bit
  - Gesamtgröße abhängig vom Granularitätsbit (G)
  - G = 0: $2^{20}$ * 1 Byte = 1 MByte
  - G = 1: $2^{20}$ * 4 Kbyte = 4 Gbyte
- P = Präsenzbit
- DPL = Descriptior Privilege Level
- Typ zur Unterscheidung verschiedene Segmenttypen
  - u.a. Code- und Datensegment
  - legt Zugriffsrechte fest

32

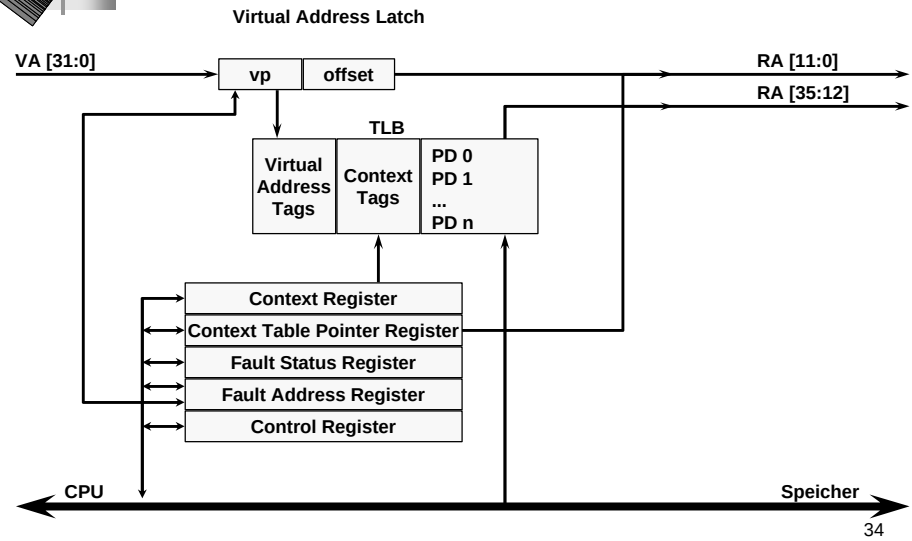
# Die SPARC MMU

- Wesentliche Eigenschaften:
  - 32 Bit virtuelle Adressen
  - 36 Bit reale Adressen
  - 4 Kbyte Seitengröße
  - 3-stufige Adreßabbildung
  - Unterstützung mehrerer Kontexte




33

# SPARC: Aufbau der MMU




34