Betriebssysteme (BS)

VL 7.2 – IA-32: Das Programmiermodell der Intel-Architektur – Protected Mode

Volkmar Sieh / **Daniel Lohmann**

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen Nürnberg

WS 20 – 7. Dezember 2020

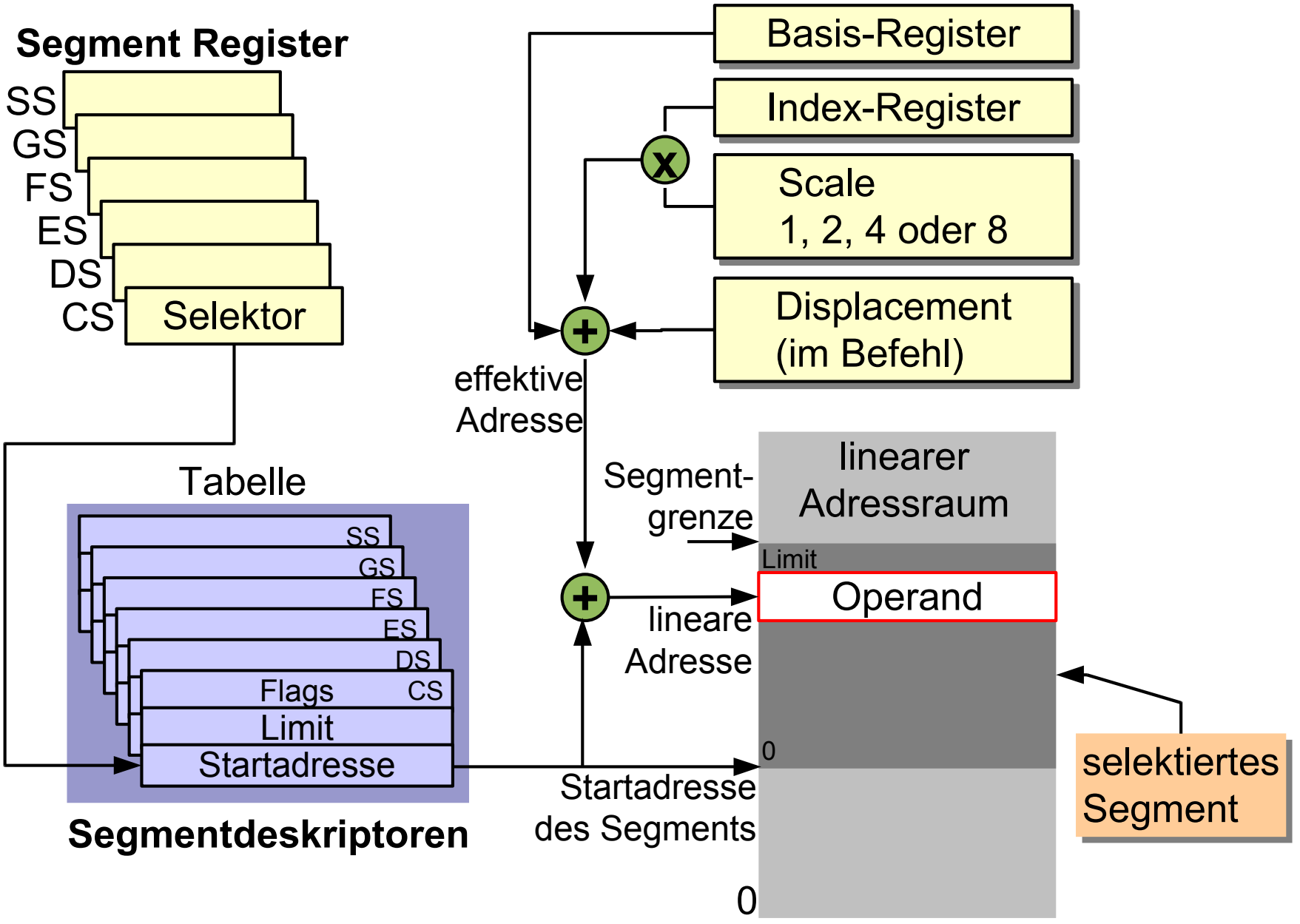
Agenda

- Einordnung
- Urvater: Der 8086
- Die 32-Bit Intel-Architektur
- Protected Mode
  - Segmente
  - MMU
  - Schutz
- Multitasking
- Zusammenfassung

IA-32: *Protected Mode* – Segmente

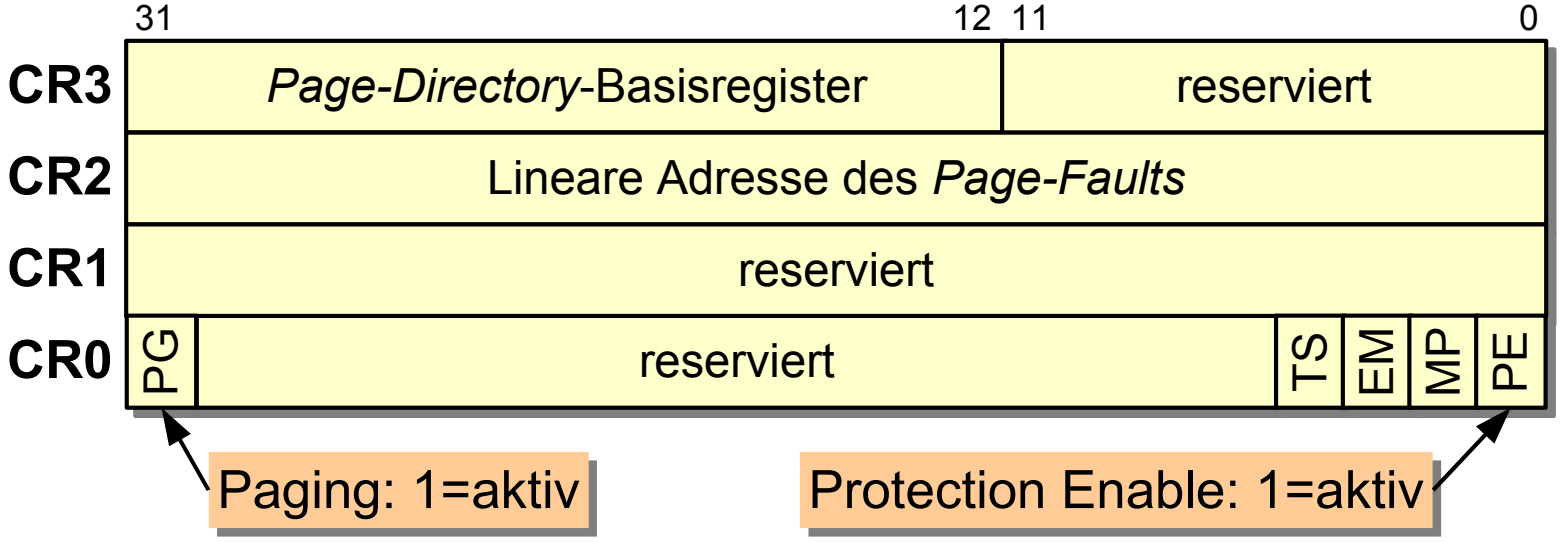
- ein Programm (in Ausführung) besteht aus mehreren Speichersegmenten
  - mindestens CODE, DATEN und STACK
  - Segmentselektoren beschreiben (indirekt) Adresse und Länge
- „Lineare Adresse“ ist Segmentstartadresse + EA
  - Segmente dürfen sich im linearen Adressraum überlappen, z.B. dürfen die Segmentstartadressen bei 0 liegen. Dadurch wird ein „flacher“ Adressraum nachgebildet.
  - „Lineare Adresse“ entspricht der physikalischen Adresse, falls die Paging Unit nicht eingeschaltet ist.

IA-32: *Protected Mode* – Segmente

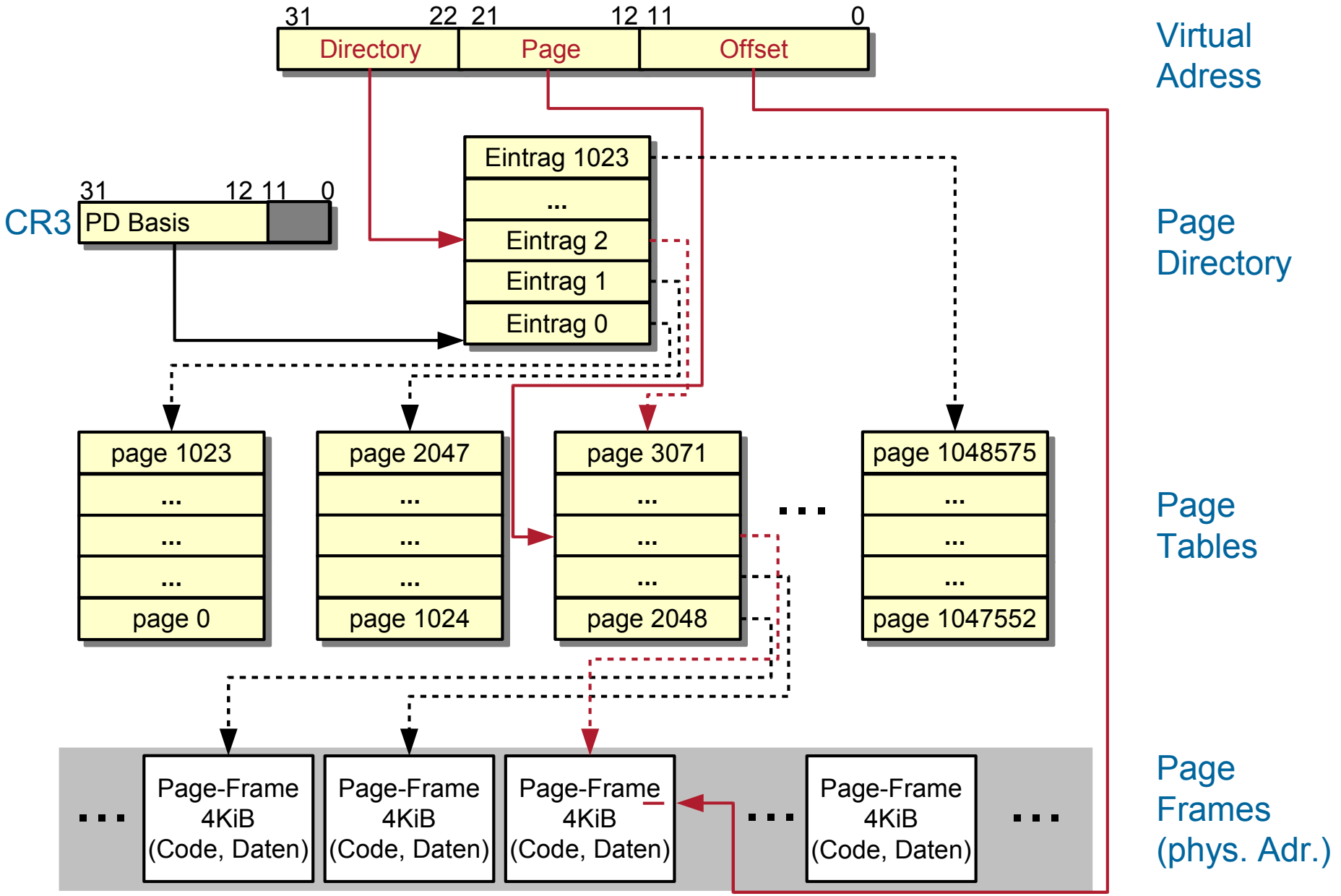


IA-32: Seitenbasierte MMU (1)

- Ein- und Auslagerung von Speicher (zwecks virtuellem Speicher) ist bei Segmentierung aufwändig. Daher bieten viele andere CPUs lediglich eine seitenbasierte Speicherverwaltung.
- ab dem 80386 kann eine ***Paging Unit*** (PU) **optional** hinzugeschaltet werden.
- die wichtigsten Verwaltungsinformationen stehen in den CRx Steuerregistern:



IA-32: Seitenbasierte MMU (2)



IA-32: TLB

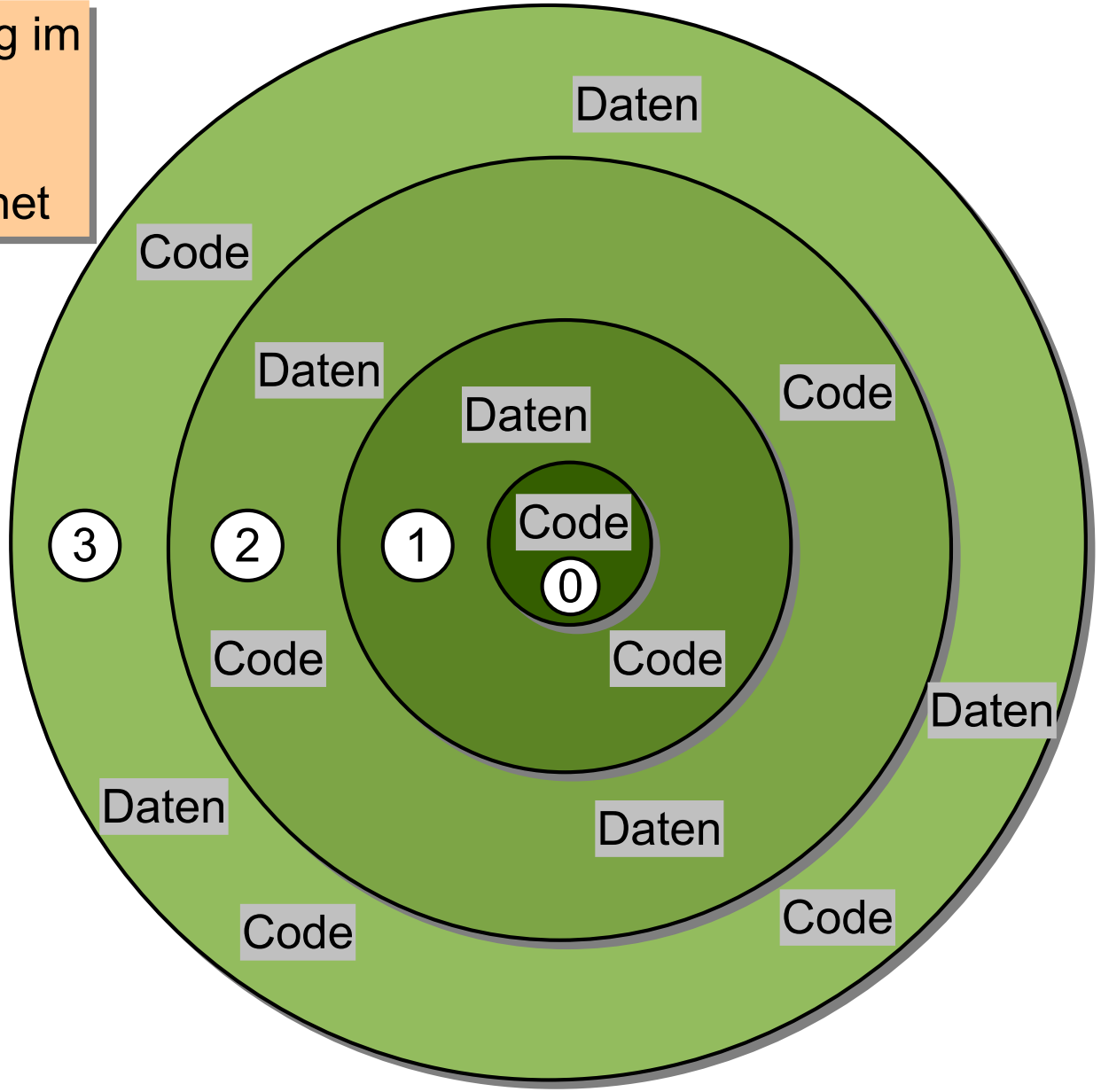
- Problem: bei aktiver *Paging Unit* wäre eine IA-32 CPU erheblich langsamer, wenn bei jedem Speicherzugriff das *Page Directory* und die *Page Table* gelesen werden müssten
- Lösung: der ***Translation Lookaside Buffer*** (TLB):
  - vollassoziativer Cache
    - Tag: 20 Bit Wert aus *Page Directory* und *Page Table* Index
    - Daten: *Page Frame* Adresse
    - Größe beim 80386: 32 Einträge
  - bei normalen Anwendungen erreicht der TLB eine Trefferrate von etwa 98%
  - Schreiben in das CR3 Register invalidiert den TLB

Schutz

- die wichtigste Eigenschaft des IA-32 Protected Mode ist das Schutzkonzept
- **Ziel:** fehlerhaften oder nicht vertrauenswürdigen Code isolieren
  - Schutz vor Systemabstürzen
  - Schutz vor unberechtigten Datenzugriffen
  - keine unberechtigten Operationen, z.B. I/O Port Zugriffe
- Voraussetzungen: Code und Daten ...
  - werden hinsichtlich der Vertrauenswürdigkeit kategorisiert
  - bekommen einen Besitzer (siehe "Multitasking")

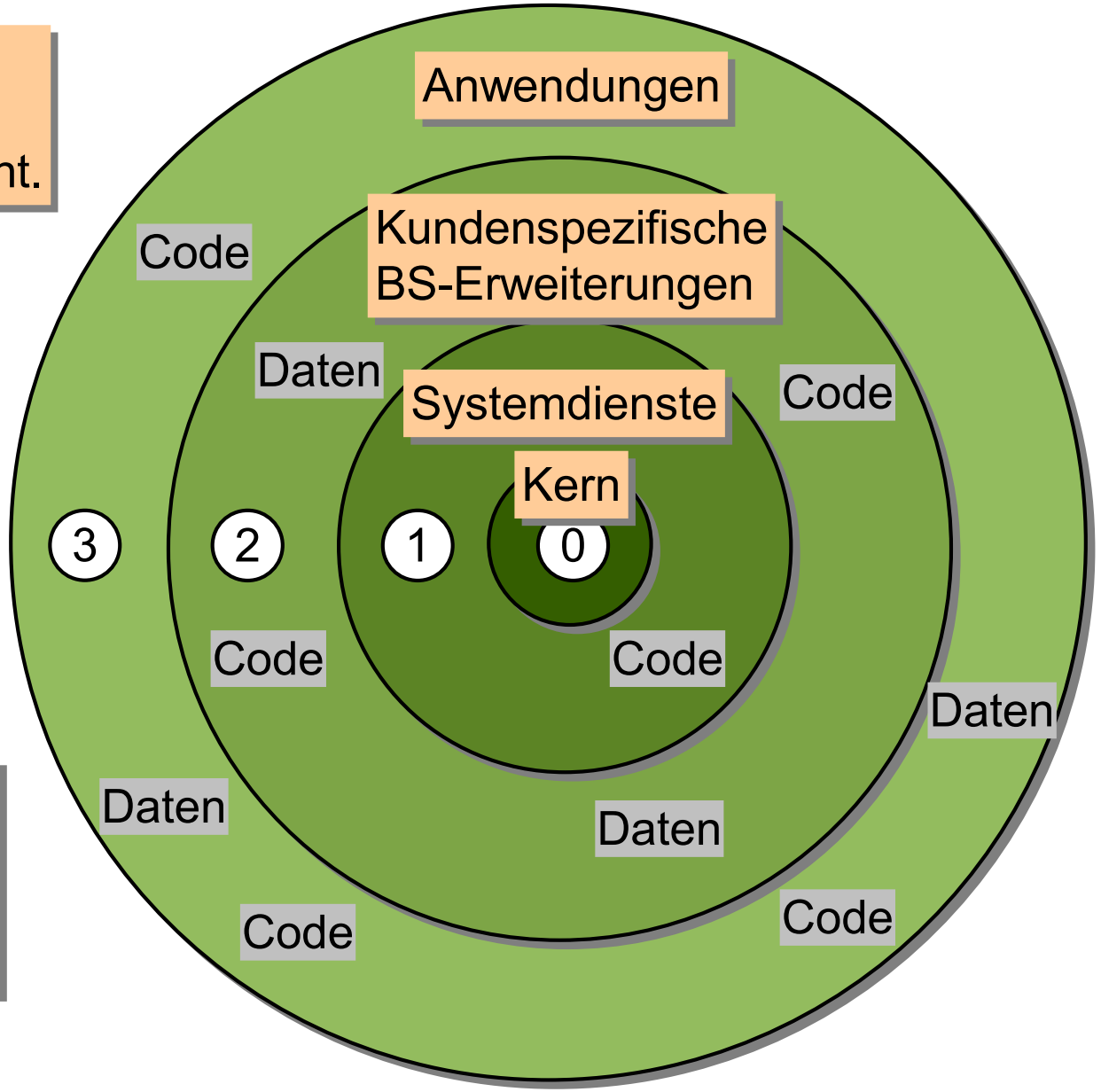
Schutzringe und *Gates*

Durch einen 2 Bit Eintrag im Segmentdeskriptor wird jedes Segment einer Privileg-Ebene zugeordnet



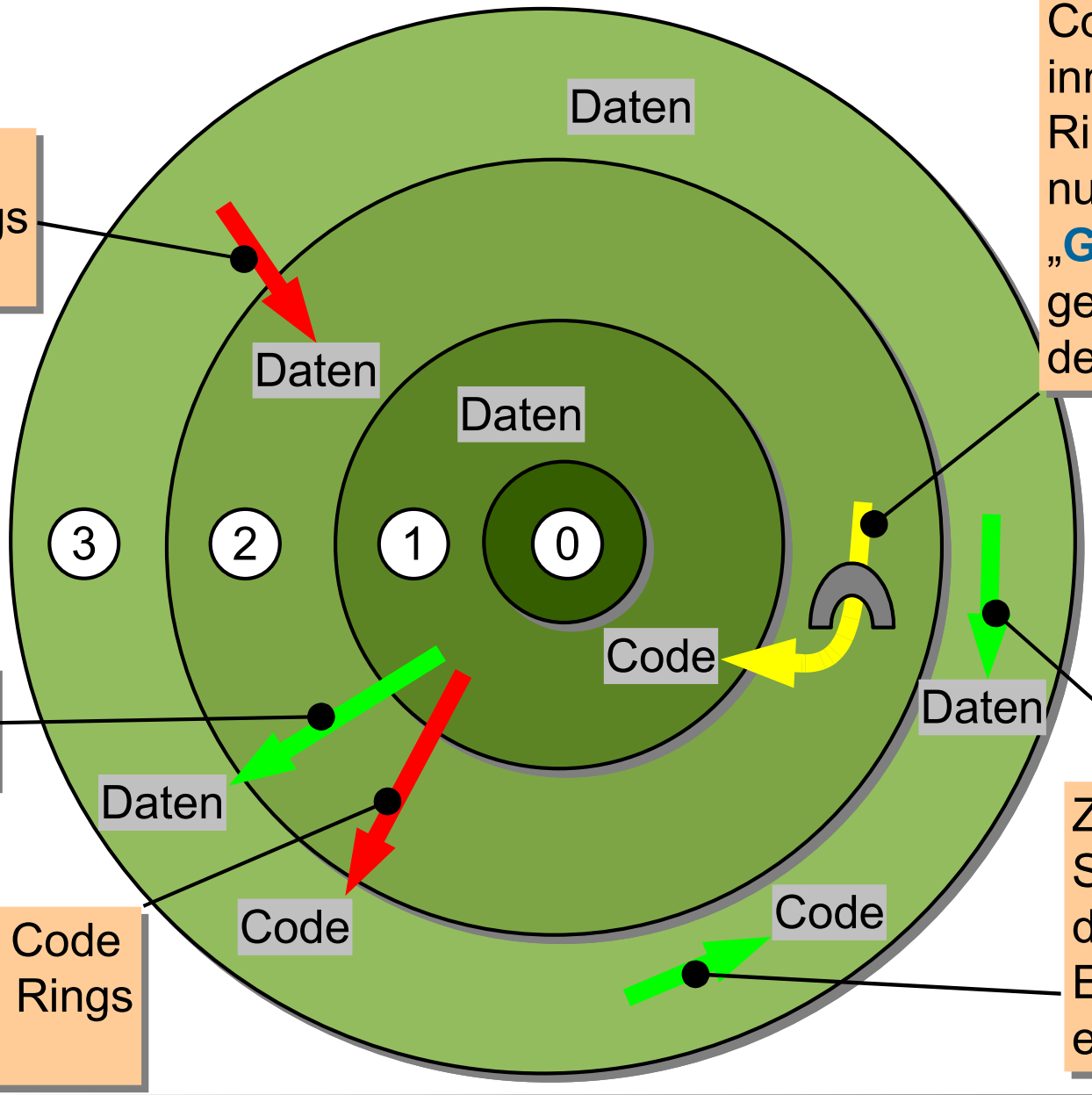
Schutzringe und *Gates*

Privileg-Ebene 3 ist die niedrigste und für Anwendungen gedacht.



Privileg-Ebene 0 ist die höchste und dem Betriebssystem*kern* vorbehalten.

Schutzringe und *Gates*



Zugriff auf Daten eines inneren Rings ist verboten

Code eines inneren Rings kann nur durch „**Gates**“ aufgerufen werden

Zugriff auf Daten eines äußeren Rings erlaubt

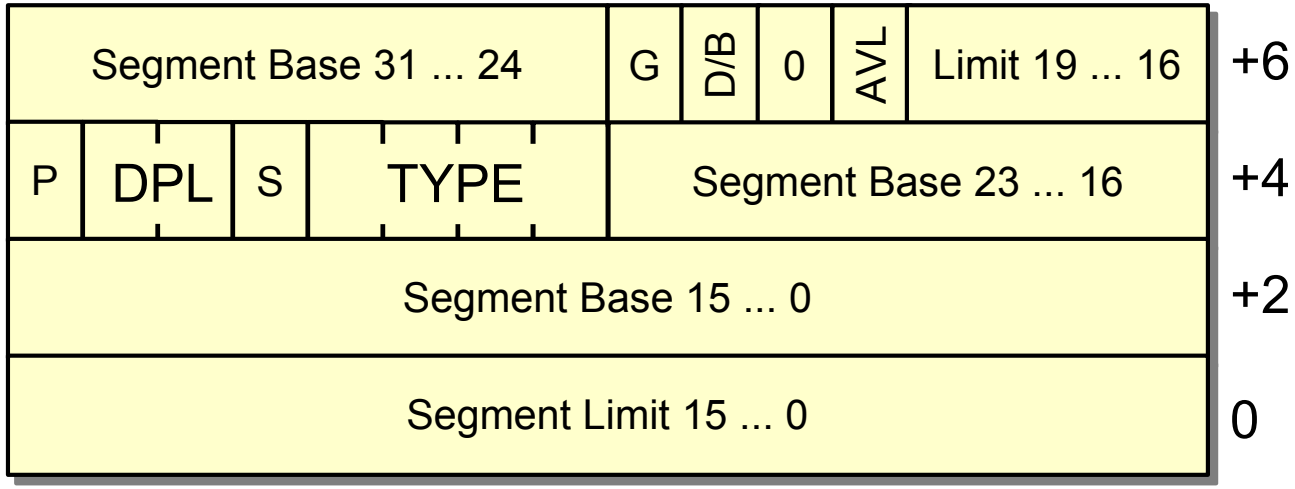
der Aufruf von Code eines äußeren Rings ist verboten!

Zugriffe auf Segmente der selben Ebene sind erlaubt

Segmentdeskriptoren

- weitere Informationen über die Schutzanforderungen der Segmente enthalten die Deskriptoren
  - jede Verletzung führt zum Auslösen einer Ausnahme

ein **Segment-Deskriptor**



TYPE – Data:

ED	- Expansion Direction
W	- Writable
A	- Accessed

TYPE – Code:

C	- Conforming
R	- Readable
A	- Accessed

P	- Present Bit
DPL	- Descriptor Privilege Level
S	- System Segment

G	- Granularity
D/B	- 16/32 Bit Seg.
AVL	- Available for OS

Beispiel: Das "flache" Speichermodell

- die meisten PC Betriebssysteme nutzen die Segmentierung nicht.
  - 32 Bit *Offset* der logischen Adresse = lineare Adresse
  - trotzdem müssen zwei Segmentdeskriptoren angelegt werden:

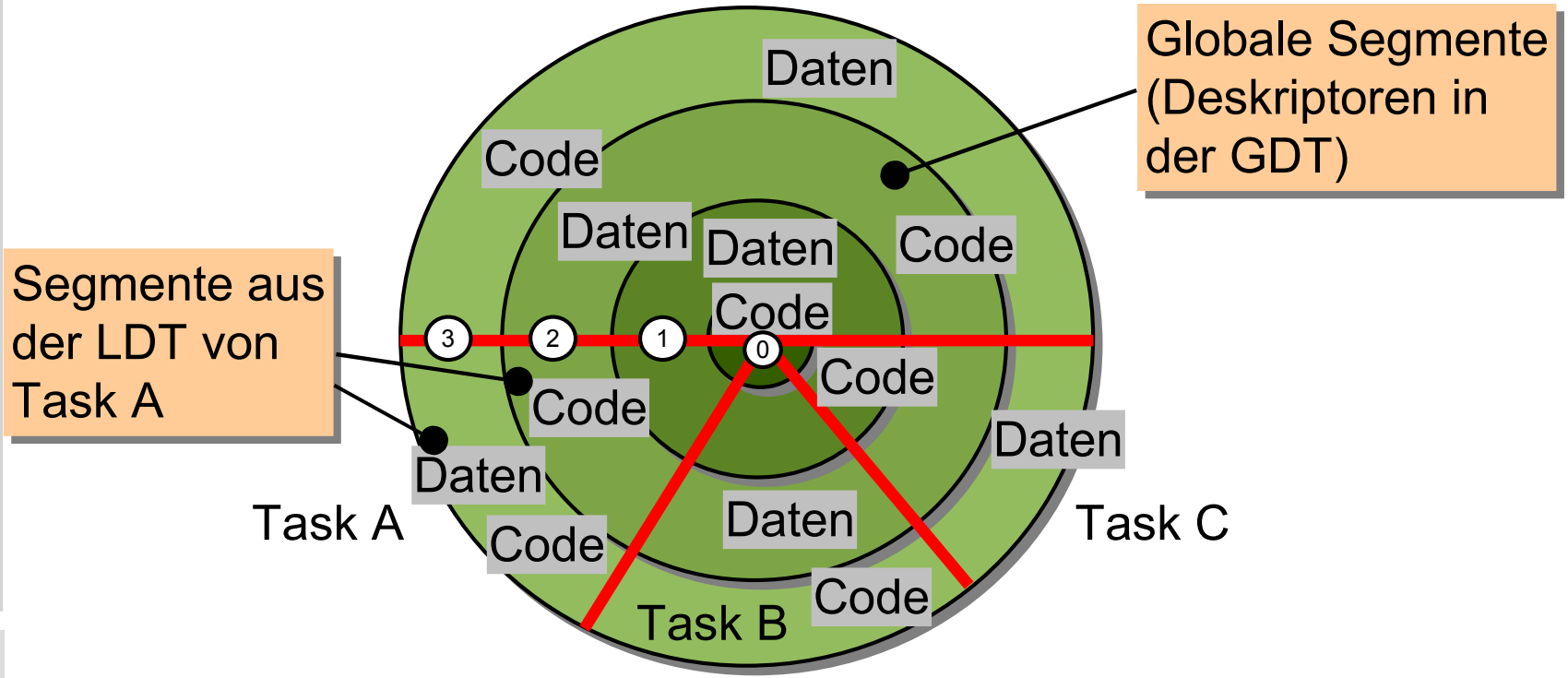
```
;
; Descriptor-Tabellen
;
gdt:
        dw      0,0,0,0   ; NULL Deskriptor

        dw      0xFFFF    ; 4Gb - (0x100000*0x1000 = 4Gb)
        dw      0x0000    ; base address=0
        dw      0x9A00    ; code read/exec
        dw      0x00CF    ; granularity=4096,
                          ; 386 (+5th nibble of limit)

        dw      0xFFFF    ; 4Gb - (0x100000*0x1000 = 4Gb)
        dw      0x0000    ; base address=0
        dw      0x9200    ; data read/write
        dw      0x00CF    ; granularity=4096,
                          ; 386 (+5th nibble of limit)
```

Agenda

Einordnung
Urvater: Der 8086
Die 32-Bit Intel-Architektur
Protected Mode
Multitasking
Zusammenfassung

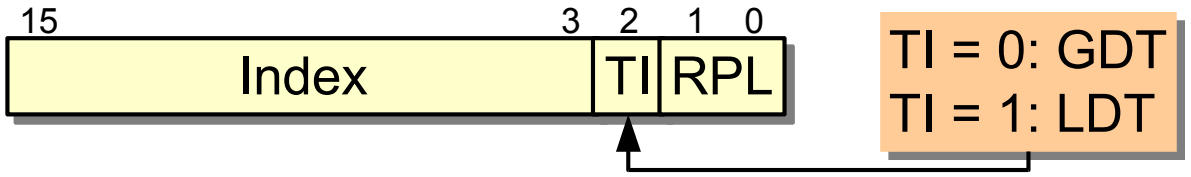
Multitasking

- neben dem Schutz vor unberechtigten "vertikalen" Zugriffen zwischen Segmenten unterschiedlicher Ebenen unterstützt IA-32 auch ein Task-Konzept ("horizontale Trennung“)
- die Zuordnung von Segmenten zu Tasks erfolgt über "Lokale Deskriptortabellen" (LDTs)



Lokale Segmentdeskriptortabellen

- ... sind nötig, wenn der Segmentselektor (z.B. aus einem Segmentregister) sich auf die LDT bezieht:



- ... werden mit Hilfe des **LDTR** gefunden, das bei jedem Taskwechsel ausgetauscht wird:

Speicherverwaltungsregister

	15 … 0	31 … 0	19 … 0
TR	TSS-Sel.	TSS-Basisadresse	TSS-Limit
LDTR	LDT-Sel.	LDT-Basisadresse	LDT-Limit
IDTR		IDT-Basisadresse	IDT-Limit
GDTR		GDT-Basisadresse	GDT-Limit

Der *Task*-Zustand: TSS Segmente

- das Task-Register TR verweist auf eine Datenstruktur, die den kompletten Task-Zustand aufnimmt
- bei einem Task-Wechsel (siehe nächste Seite) wird der komplette Zustand gesichert und der Zustand des Ziel-Tasks geladen
  - alles in Hardware!

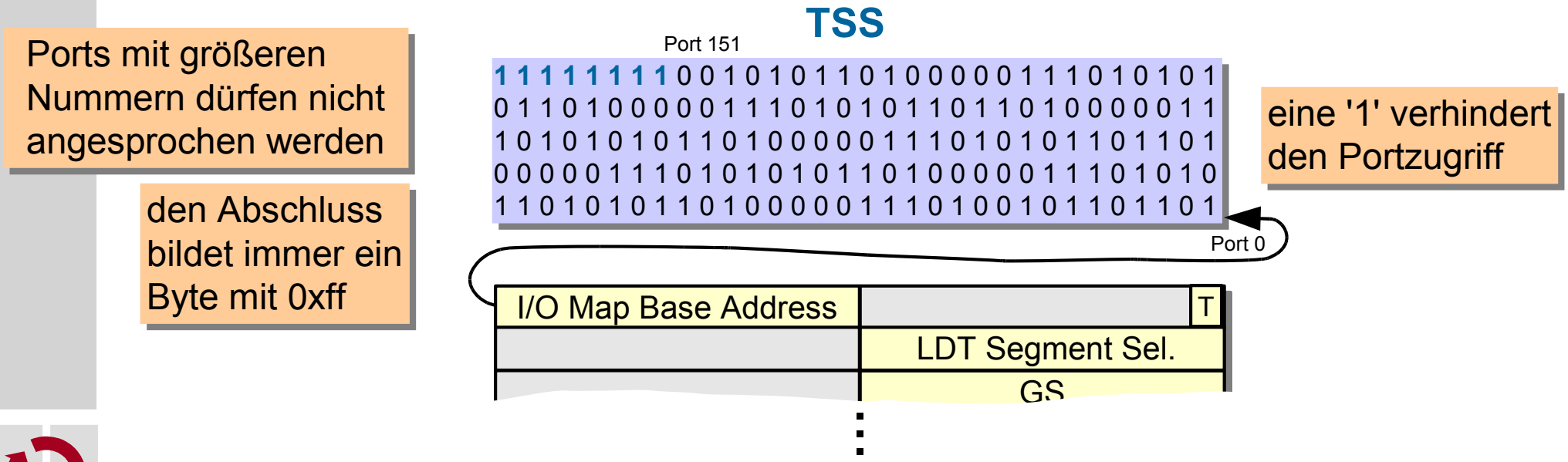
I/O Map Base Address	T
	LDT Segment Sel.
	GS
	FS
	DS
	SS
	CS
	ES
EDI	
ESI	
EBP	
ESP	
EBX	
EDX	
ECX	
EAX	
EFLAGS	
EIP	
CR3 (PDBR)	
	SS2
ESP2	
	SS1
ESP1	
	SS0
ESP0	
	Prev. Task Link

Task-Wechsel

- für einen Task-Wechsel benötigt man entweder ...
  - ein Task-Gate in der GDT, einer LDT oder der IDT (Task-Wechsel bei Unterbrechungen!)
  - oder einfach nur einen TSS Deskriptor in der GDT
- ausgelöst werden kann ein Wechsel durch ...
  - eine JMP Instruktion
  - eine CALL Instruktion
  - eine Unterbrechung
  - eine IRET Instruktion
- Nested Tasks
  - bei Unterbrechungen und CALLs wird das NT Flag im EFLAGS Register und der "Prev. Task Link" im TSS gesetzt.
  - Wenn dies der Fall ist, springt IRET zum vorherigen Task zurück.

Ein-/Ausgaben im Protected Mode

- nicht jeder beliebige Task darf Ein-/Ausgabe durchführen!
- Zugriffe auf Geräte im Speicher (memory-mapped I/O) können über Speicherschutz abgefangen werden
- Zugriffe auf I/O Ports werden eingeschränkt:
  - die I/O Privilege Level Bits im EFLAGS Register erlauben Ein- und Ausgaben auf bestimmten Schutzringen
  - auf den anderen Ebenen regelt die I/O Permission Bitmap für jeden Task und Port der Zugriff:



IA-32: Was gibt es sonst noch?

- Physical Address Extension (PAE)
  - ab Pentium Pro: 36-Bit Adressen (physikalisch)
  - erweiterte *Page Table* Einträge
  - weitere *Page Directory* Ebene
- System Management Mode (SMM)
  - gibt dem BIOS Kontrolle über das System
  - das Betriebssystem merkt davon nichts!
- Virtualisierung der CPU
  - der Virtual 8086 Mode
    - 16 Bit Anwendungen oder Betriebssysteme laufen als IA-32 Task in einer geschützten Umgebung
  - Vanderpool Technology
    - Hardwareunterstützung für virtuelle Maschinenlösungen wie VmWare, VirtualPC oder Xen
    - erlaubt die Ausführung von E0 Protected Mode Code in einer VM

Agenda

Einordnung
Urvater: Der 8086
Die 32-Bit Intel-Architektur
Protected Mode
Multitasking
Zusammenfassung

Zusammenfassung

- die IA-32 Architektur ist ausgesprochen komplex
  - segmentbasierter und seitenbasierter Speicherschutz
  - Hardwareunterstützung für Multitasking
  - Task-Aktivierung bei Unterbrechungen
  - Schutz von I/O Ports pro Task
  - ...
- viele dieser Features werden von heutigen Betriebssystemen nicht genutzt
  - typisch ist der flache Adressraum mit aktiver Paging Unit
  - Hardware-Tasks sind kaum portierbar
- bemerkenswert ist auch die konsequente Kompatibilität mit älteren Prozessorversionen
  - Stichwort "PIC" und "A20 Gate"! 

Zusammenfassung

- die
  - se
  - H
  - Ta
  - S
  - ...
- viel
  Bet
  - ty
  - H
- ber
  mit
  - Stichwort "PIC" und "A20 Gate"!

Was ist mit 64-Bit? ⇝ AMD64 (x86-64)

- 64-Bit-Erweiterung von x86, propagiert von AMD
  - offizielle Intel-Bezeichnung: x86-64
  - **Achtung:** IA-64 bezeichnet Intels (tote?) Itanium-Architektur
- Wesentliche Features
  - 64-Bit-Register und -Adressen
  - 16 (statt bisher 8) General-Purpose Register
- Viele IA-32-Features sind weggefallen!
  - Segmentierung (außer FS und GS Register)
    ⇝ MMU *muss* benutzt werden
  - Task-Modell
  - Ring-Modell (außer -1, 0, 3)
  - Virtual-X86-Modus