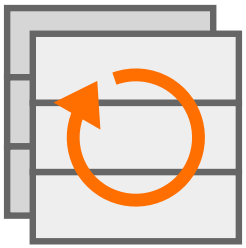


TUHH
Technische Universität Hamburg

Betriebssystembau (BSB)

VL 6 – IA-32: Das Programmiermodell der Intel-Architektur

Christian Dietrich

Operating System Group

SS 22 – 03. Mai 2021

Interaktion während der Veranstaltung

TUHH
Technische Universität Hamburg

- BSB ist vom "Stil" her eine **interaktive Präsenzveranstaltung**
  - Wir wollen versuchen, dieses soweit wie möglich "online" zu retten

⇝ **Synchrones** Format – Fragen und Beteiligung ist erwünscht!

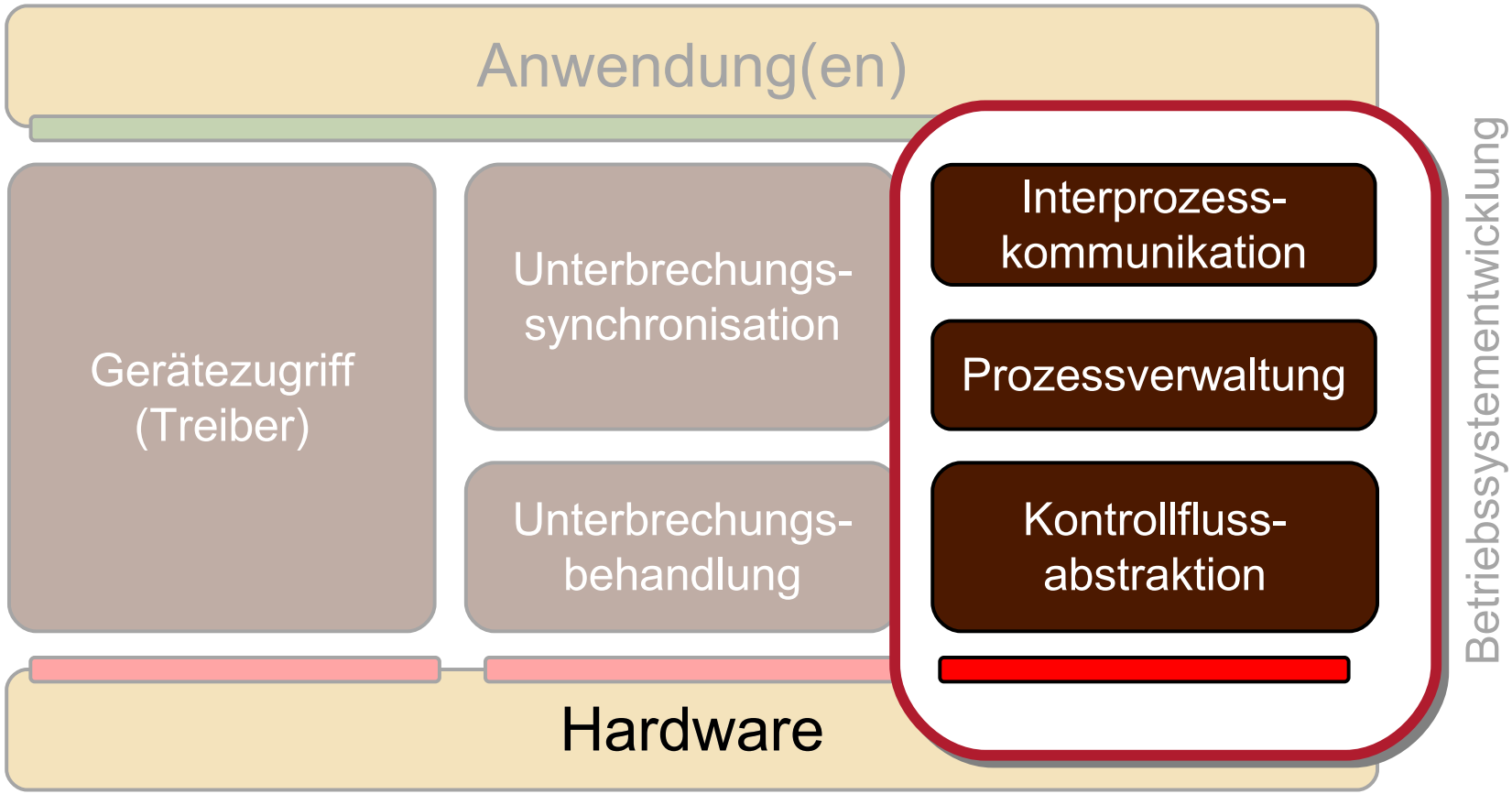
- Interaktion **während** der Veranstaltung
  - 1. „Melden“
  - 2. „Drankommen“
  - 3. Profit
- Interaktion **außerhalb** der Veranstaltung
  - Über das Stud.IP-Forum
  - **NEU**: EIM Mattermost Team: `https://communicating.tuhh.de/eim`

Aufzeichnung der Vorlesung

You are on mic!

- Auf vielfachen Studierendenwunsch:
  Veranstaltung wird aufgezeichnet
  - Wird im Anschluss über Stud.IP verfügbar gemacht
  - ⇝ Geschlossene Nutzergruppe
- Aufgezeichnet wird
  - Screencast der BBB-Session **ohne den Chat (Klarnamen)**
  - Ihre Stimme bei Fragen und Anmerkungen
  - **Durch Aktivierung Ihres Mikrofons willigen Sie dazu ein!**
- Fragen können über direkte Nachricht an mich auch anonym gestellt werden

Überblick dieser VL



Agenda

TUHH
Technische Universität Hamburg

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

Historie der Intel x86-Prozessoren

TUHH
Technische Universität Hamburg

- 1978: **8086** der Urvater des PC Prozessors
- 1982: **80286** Einführung des Protected Mode
  - segmentbasierter Speicherschutz
- 1985: **80386** erster IA-32 Prozessor
  - seitenbasierter virtueller Speicher
  - Protected Mode
- 1989: **80486** integrierte FPU, RISC Ansätze
- 1993: **Pentium** P5-Architektur
  - superskalar, 64-Bit Datenbus
  - SMM, MMX, APIC, Dualprozessor-fähig

Historie der Intel x86-Prozessoren (Forts.)

TUHH
Technische Universität Hamburg

- 1995: **Pentium Pro** P6-Architektur
  - 36-Bit Adressbus (PAE)
  - Level 2 Cache „on chip“, RISC-artige Mikroinstruktionen
- 1997: **Pentium II** Pentium Pro + MMX
  - Level 2 Cache wieder extern, dafür bessere 16-Bit Performance
- 1999: **Pentium III** SSE, Pentium M (2003)
- 2000: **P4** Netburst-Architektur
  - SSE2, optimiert für hohe Taktzahlen (angedacht bis zu 10 GHz)
- 2004: **P4 Prescott** Erweiterte Netburst Architektur
  - Hyperthreading, Vanderpool, EM64T, 31-stufige Pipeline!

Historie der Intel x86-Prozessoren (Forts.)

TUHH
Technische Universität Hamburg

- 2005: **Core** Ende der Netburst Architektur
  - geringerer Takt, weniger Strom, aber bessere Performance!
  - Architektur basiert auf P6-Architektur, kein Hyperthreading
- 2006: **Core 2** Dual Core, Quad Core, 64 Bit
- 2009: **Core i7** Nehalem . . . Whiskey Lake (2018)
  - Hyperthreading, Octa Core, Quick Path Interconnect
  - “Power Control Unit” (PCU) passt Takt der TDP an
  - Unterstützung für “Hardware-Transactional Memory” (HTM)
- 2019: **Core i7** Sunny Cove . . .
  - Besserer Schutz vor μ-arch exploits (Meltdown/Spectre)
- 2012: **Xeon Phi** Larrabee . . . Knights Mill (2017)
  - P54C Manycore (62 cores), SIMD Instruktionen
- 2008: **Atom** extrem stromsparend
  - Architektur (wieder) CISC-lastiger, Ähnlichkeiten mit 486/Pentium

Agenda

TUHH
Technische Universität Hamburg

8086: Programmiermodell

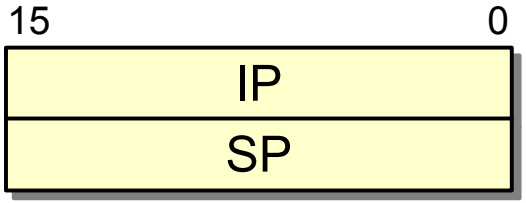
TUHH
Technische Universität Hamburg

- 16-Bit Architektur, little-endian
- 20-Bit Adressbus, d.h. maximal 1 MiB Hauptspeicher
- wenige Register
  - (jedenfalls aus heutiger Sicht)
- 123 Befehle
  - kein orthogonaler Befehlssatz
- Befehlslängen von 1 bis 4 Byte
- segmentierter Speicher
- **noch immer aktuell**
  - obwohl von 1978 noch heute von jeder IA-32 CPU unterstützt
    - *Real Mode*, *Virtual 8086 Mode*

Aufwärtskompatibilität wird bei Intel groß geschrieben

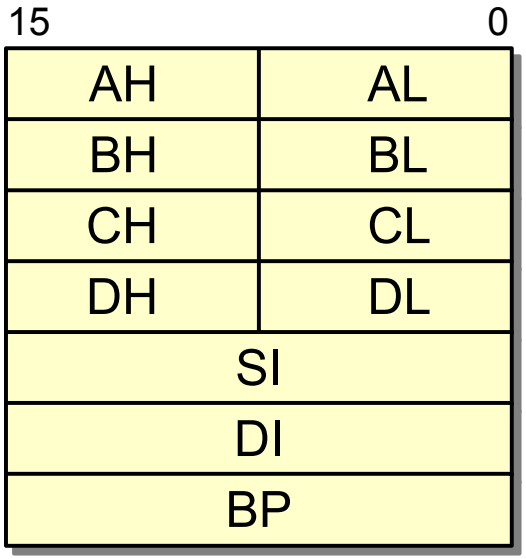
8086: Registersatz

Befehls- und Stapelzeiger

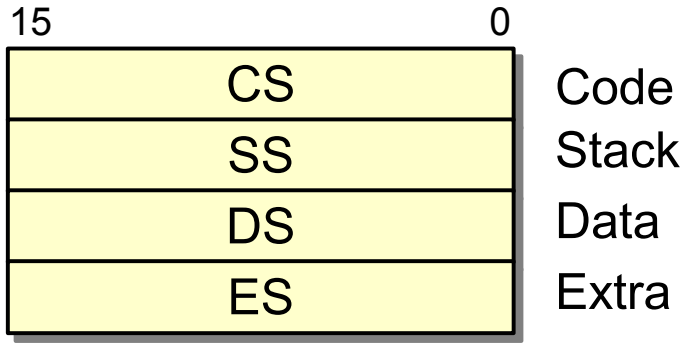


Flag Register

Vielzweckregister

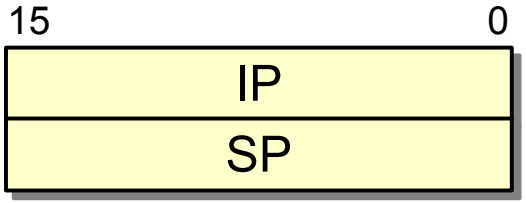


Segmentregister

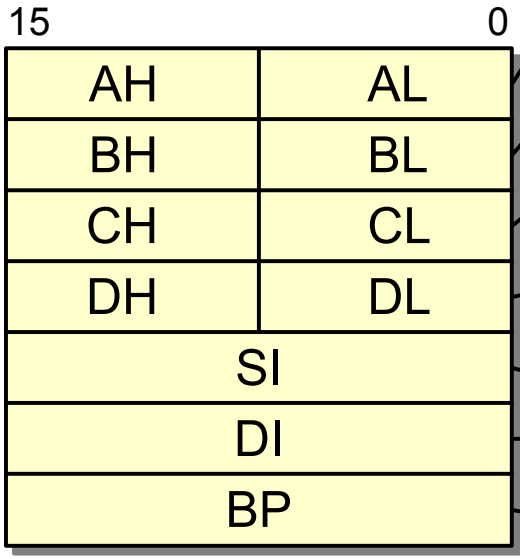


8086: Registersatz

Befehls- und Stapelzeiger



Vielzweckregister



AX: Accumulator Register
- arithmetisch-logische Operationen
- I/O
- kürzester Maschinencode

BX: Base Address Register

CX: Count Register
- für LOOP Befehl
- für *String* Operationen mit REP
- für Bit *Shift* und *Rotate*

DX: Data Register
- DX:AX sind 32 Bit für MUL/DIV
- Portnummer für IN und OUT

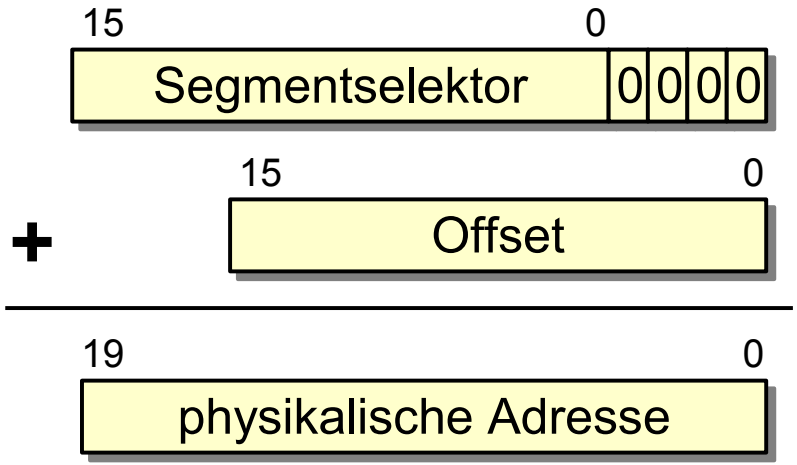
SI, DI: Index Register
- für Array-Zugriffe (Displacement)

BP: Base Pointer

Jedes „Vielzweckregister“ erfüllt seinen speziellen Zweck

8086: Segmentierter Speicher

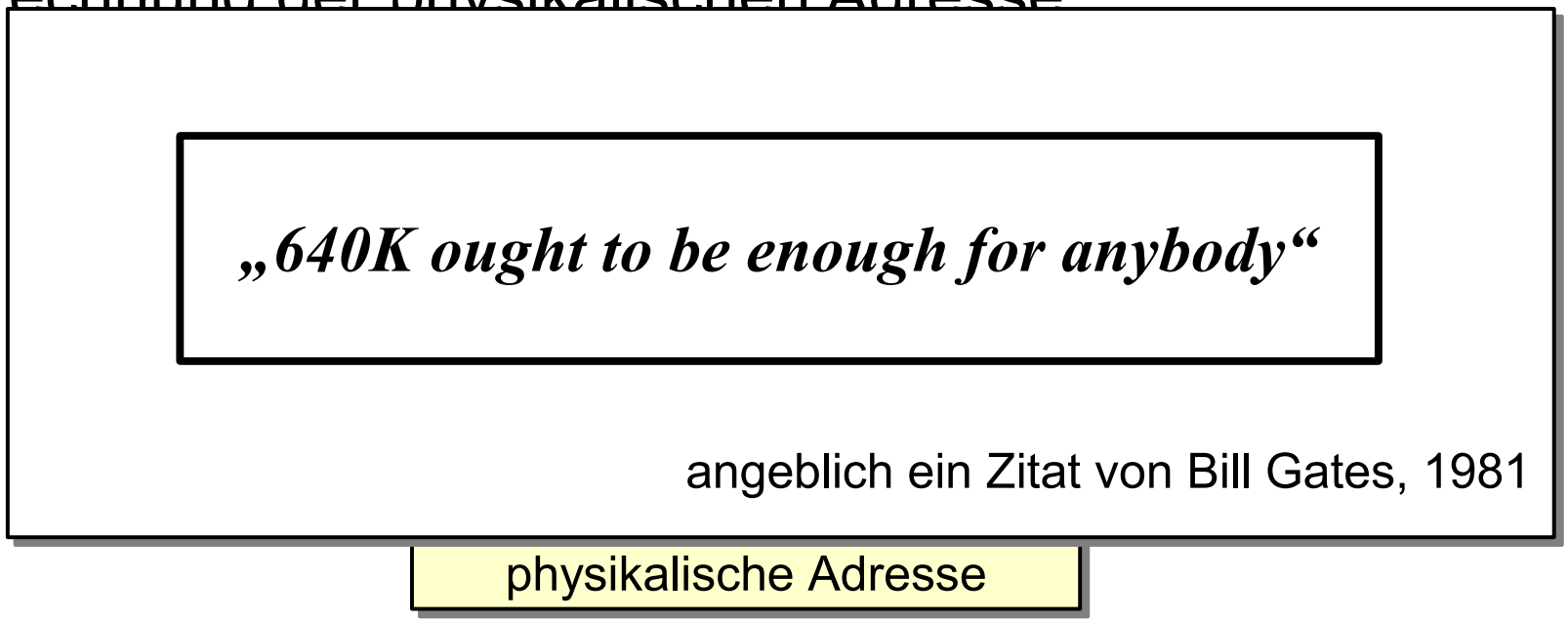
- logische Adressen bestehen beim 8086 aus
  - Segmentselektor (i.d.R. der Inhalt eines Segmentregisters)
  - Offset (i.d.R. aus einem Vielzweckregister oder dem Befehl)
- Berechnung der physikalischen Adresse:



die 16 Bit Konkurrenz konnte i.d.R. nur 64KB adressieren

8086: Segmentierter Speicher

- logische Adressen bestehen beim 8086 aus
  - Segmentselektor (i.d.R. der Inhalt eines Segmentregisters)
  - Offset (i.d.R. aus einem Vielzweckregister oder dem Befehl)
- Berechnung der physikalischen Adresse:



die 16 Bit Konkurrenz konnte i.d.R. nur 64KB adressieren

8086: Speichermodelle

TUHH
Technische Universität Hamburg

- Programme können Adressen unterschiedlich bilden. Das Ergebnis waren unterschiedliche Speichermodelle:
  - **Tiny**
    - Code-, Daten- und Stacksegment sind identisch: 64K insgesamt
  - **Small**
    - Trennung des Codes von Daten und Stack: 64K + 64K
  - **Medium**
    - 32(20) Bit Zeiger für Code, Daten- und Stapelseg. aber fest (64K)
  - **Compact**
    - Festes Code Segment (64K), 32(20) Bit Zeiger für Daten und Stack
  - **Large**
    - „far“ Zeiger für alles: 1MB komplett nutzbar
  - **Huge**
    - wie „Large“, aber mit normalisierten Zeigern

8086: Fazit

TUHH
Technische Universität Hamburg

- Urvater der PC-Prozessoren
  - bildete den Kern der ersten PCs
  - noch heute sind IA32-Prozessoren kompatibel
- Segmentregister brachten Vorteile
  - trotz 16-Bit-Architektur 1 MB Speicher
  - Trennung von logischen Modulen im Hauptspeicher
- Programm- und Übersetzerentwicklung ist aber vergleichsweise schwierig
  - verschiedene Speichermodelle
  - nicht orthogonaler Befehlssatz

Agenda

TUHH
Technische Universität Hamburg

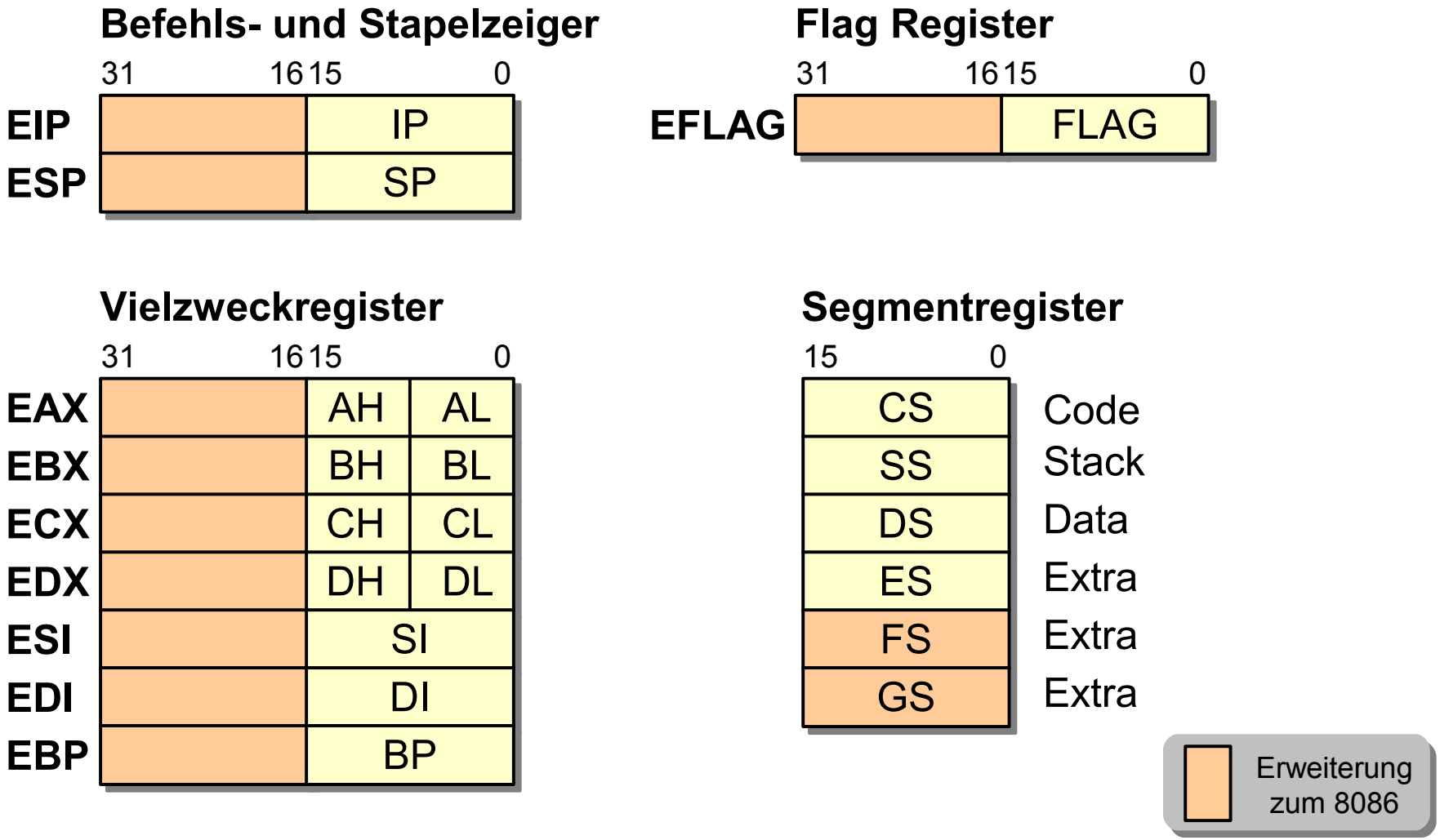
IA-32 – die 32 Bit Intel Architektur

TUHH
Technische Universität Hamburg

- die erste IA-32 CPU war der **80386**
  - wobei der Begriff „IA-32“ erst sehr viel später eingeführt wurde
- 32 Bit Technologie: Register, Daten- und Adressbus
  - ab Pentium Pro: 64 Bit Daten und 36 Bit Adressbus
- zusätzliche Register
- komplexe Schutz- und Multitaskingunterstützung
  - *Protected Mode*
  - ursprünglich schon mit dem 80286 (16-Bit) eingeführt
- Kompatibilität
  - mit älteren Betriebssystemen durch den *Real Mode*
  - mit älteren Anwendungen durch den *Virtual 8086 Mode*
- segmentbasiertes Programmiermodell
- seitenbasierte MMU

80386: Registersatz (Erweiterung)

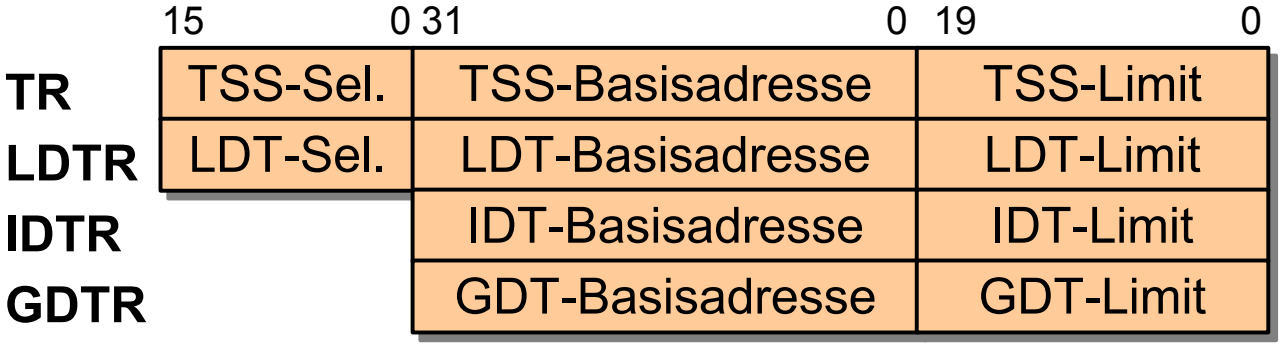
- erweiterte Register heißen aus Kompatibilitätsgründen E...



80386: Registersatz (neue Register)

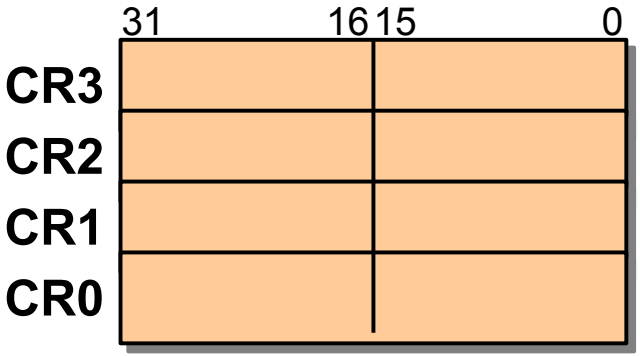
TUHH
Technische Universität Hamburg

Speicherverwaltungsregister

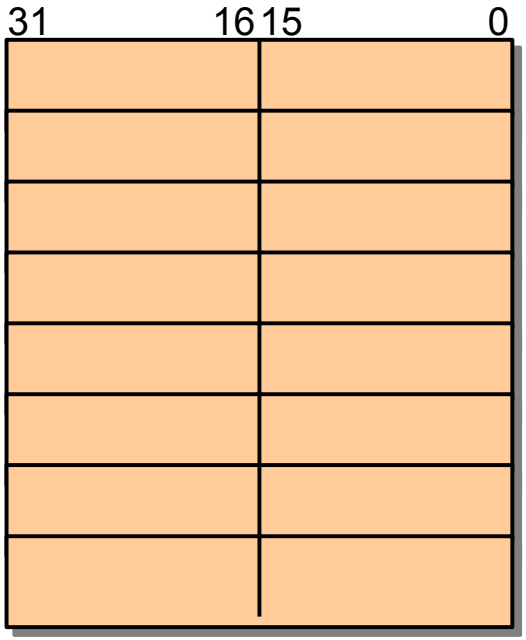


Erläuterungen folgen ...

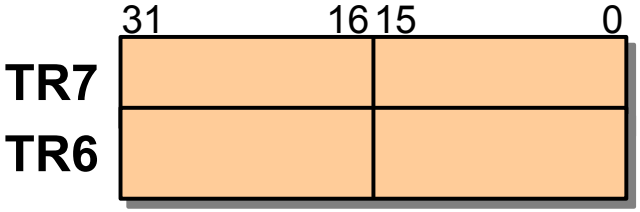
Steuerregister



Debugregister

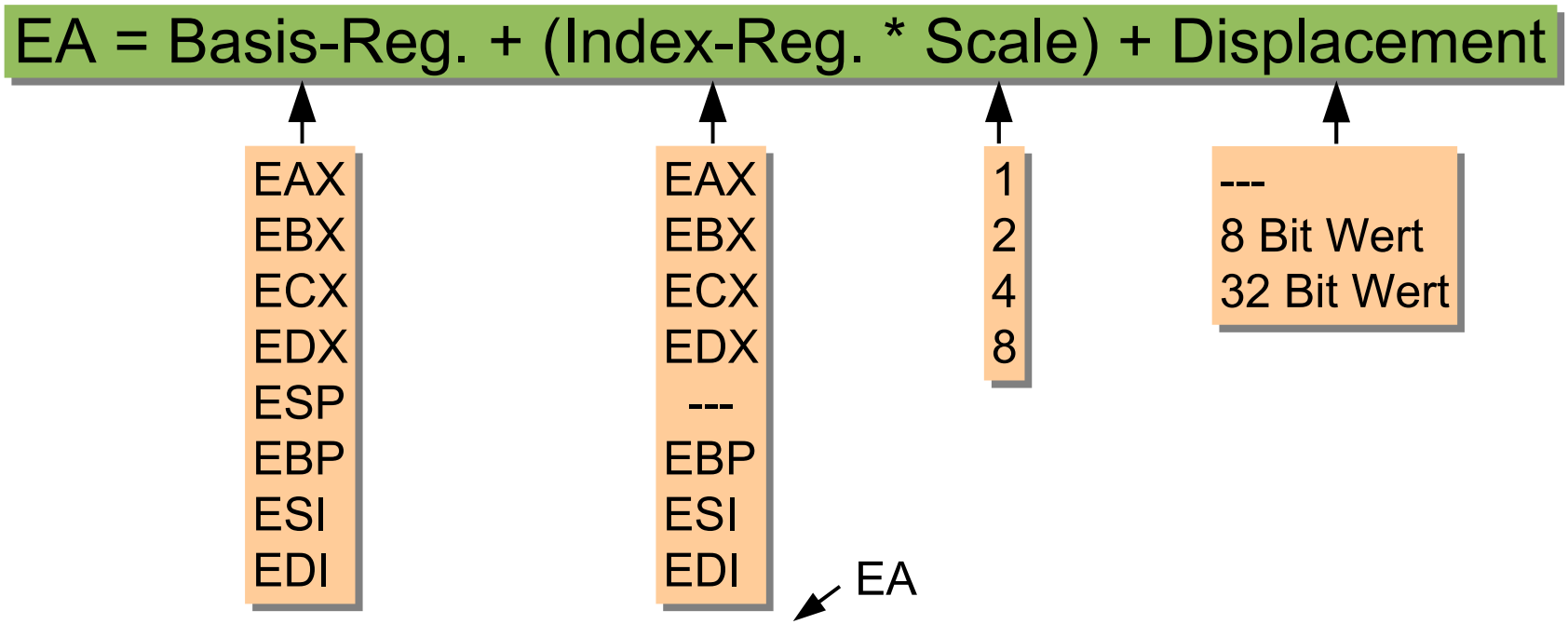


Testregister



IA-32: Adressierungsarten

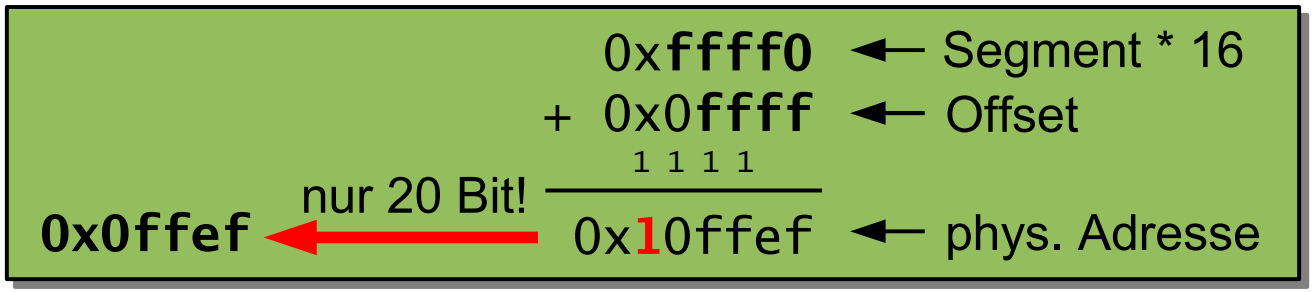
- Effektive Adressen (EA) werden nach einem einfachen Schema gebildet
  - alle Vielzweckregister können dabei gleichwertig verwendet werden



- Beispiel: MOV EAX, **Feld[ESI * 4]**
  - Lesen aus Feld mit 4 Byte großen Elementen und ESI als Index

Das A20-Gate

- ... ist ein Relikt aus der Zeit der 80286 Systeme (IBM AT)
  - beim IBM XT (8086) konnte es bei der Adressberechnung zu einem Überlauf kommen. Im Maximalfall:



  - MS-DOS (und andere Systeme) verwenden diesen „Trick“.
  - Aus Kompatibilitätsgründen wurde im IBM AT die A20-Leitung über das „A20 Gate“ (Register im **Tastaturcontroller**) maskiert.
    - A20 muss explizit freigeschaltet werden, um Speicher > 1 MiB zu addressieren
  - Ab dem 486 hat Intel das A20-Gate in die CPU integriert!

Agenda

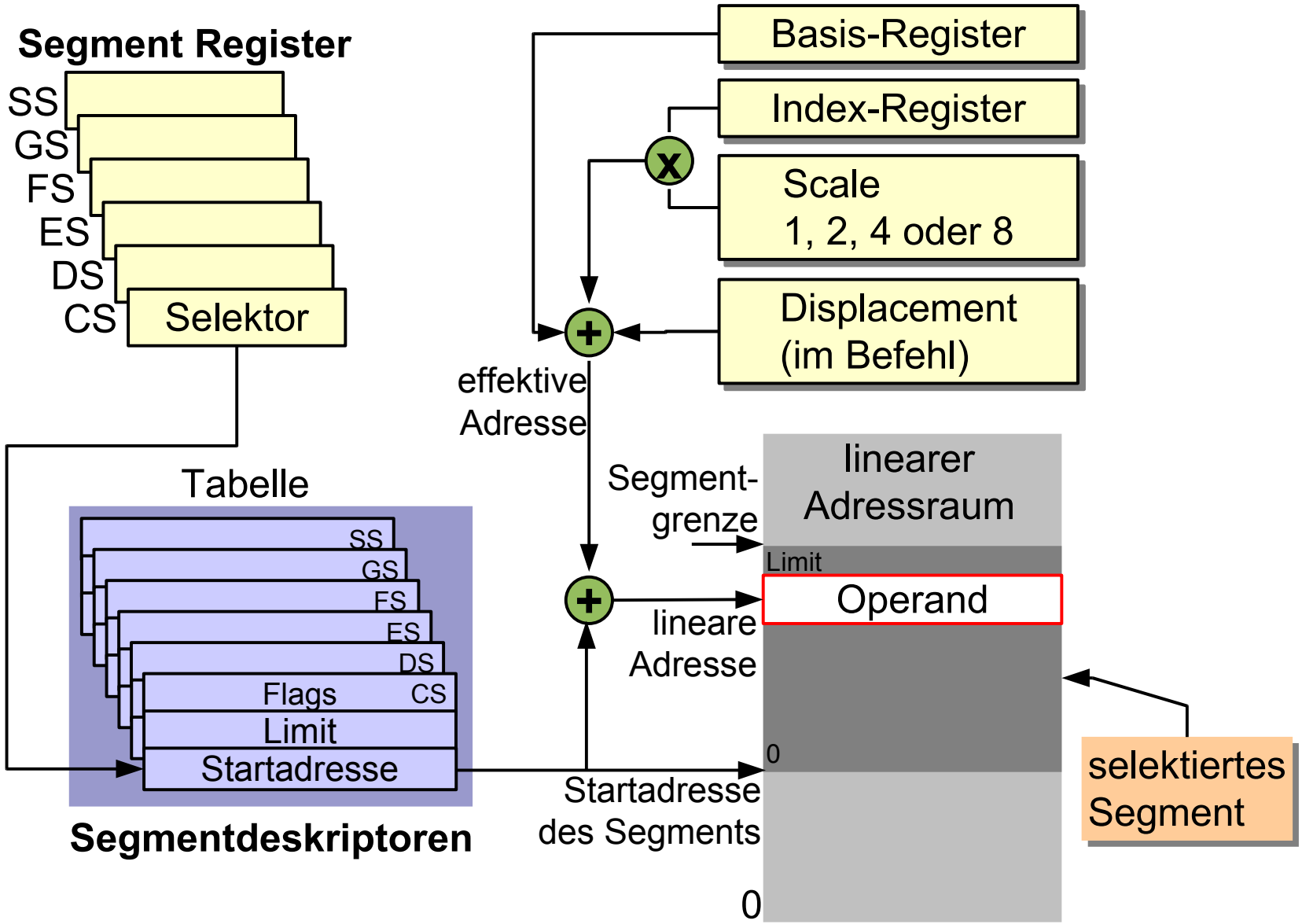
TUHH
Technische Universität Hamburg

IA-32: Protected Mode - Segmente

TUHH
Technische Universität Hamburg

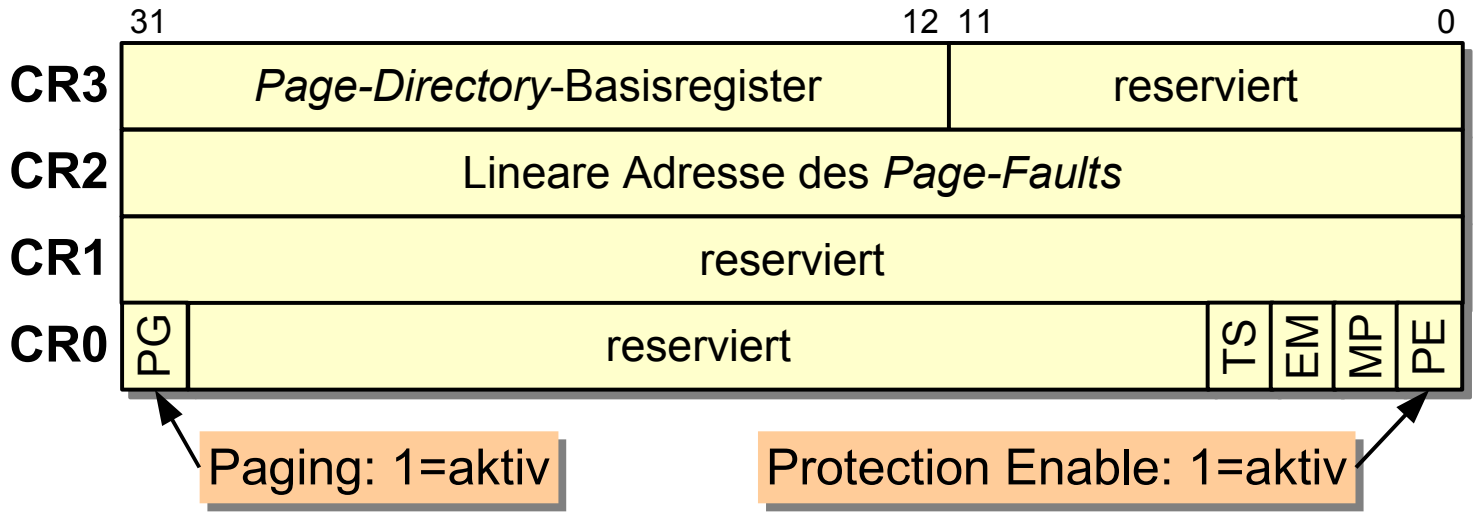
- ein Programm (in Ausführung) besteht aus mehreren Speichersegmenten
  - mindestens CODE, DATEN und STACK
  - Segmentselektoren beschreiben (indirekt) Adresse und Länge
- „Lineare Adresse“ ist Segmentstartadresse + EA
  - Segmente dürfen sich im linearen Adressraum überlappen, z.B. dürfen die Segmentstartadressen 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

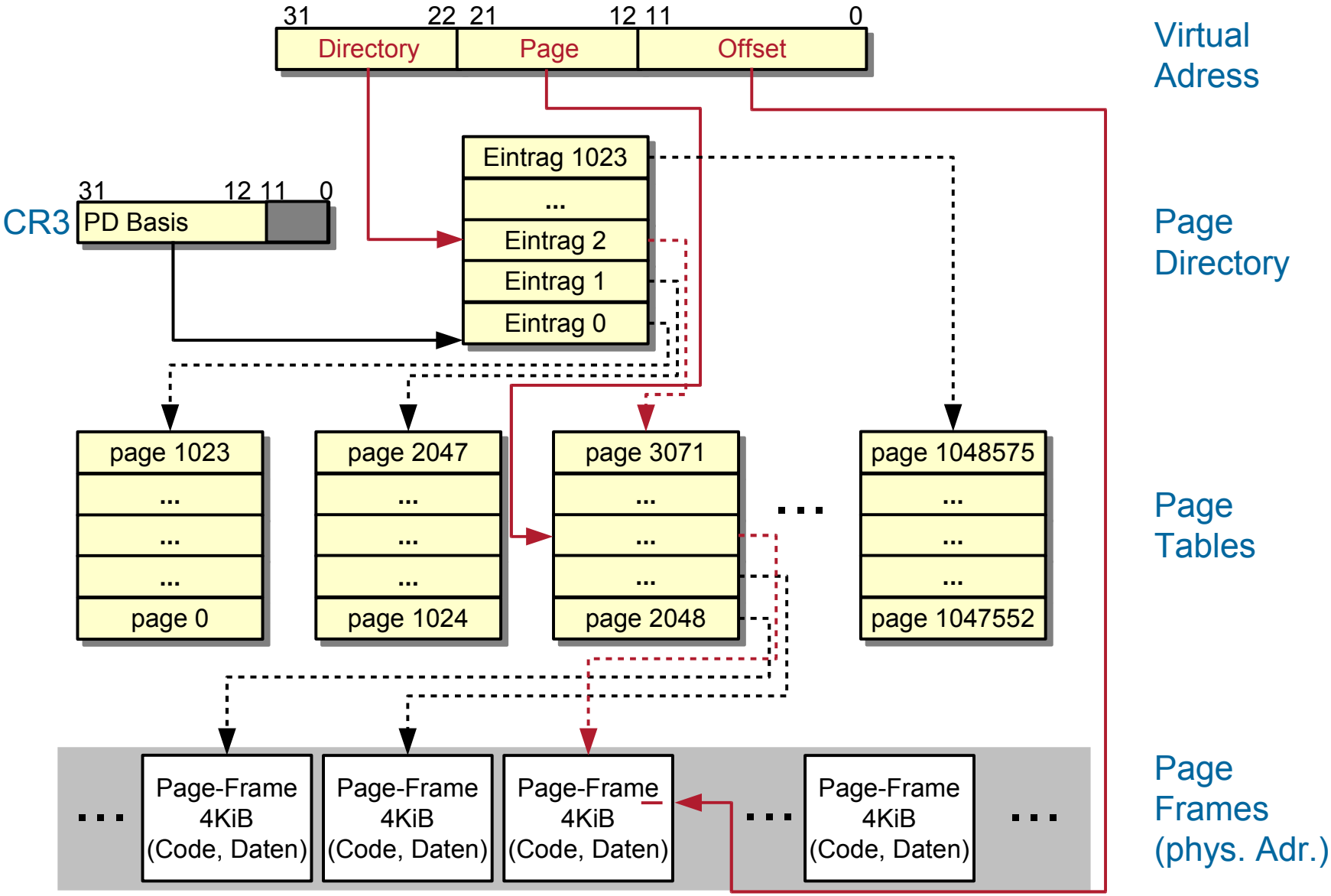


IA-32: Seitenbasierte MMU (1)

- 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

TUHH
Technische Universität Hamburg

- 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

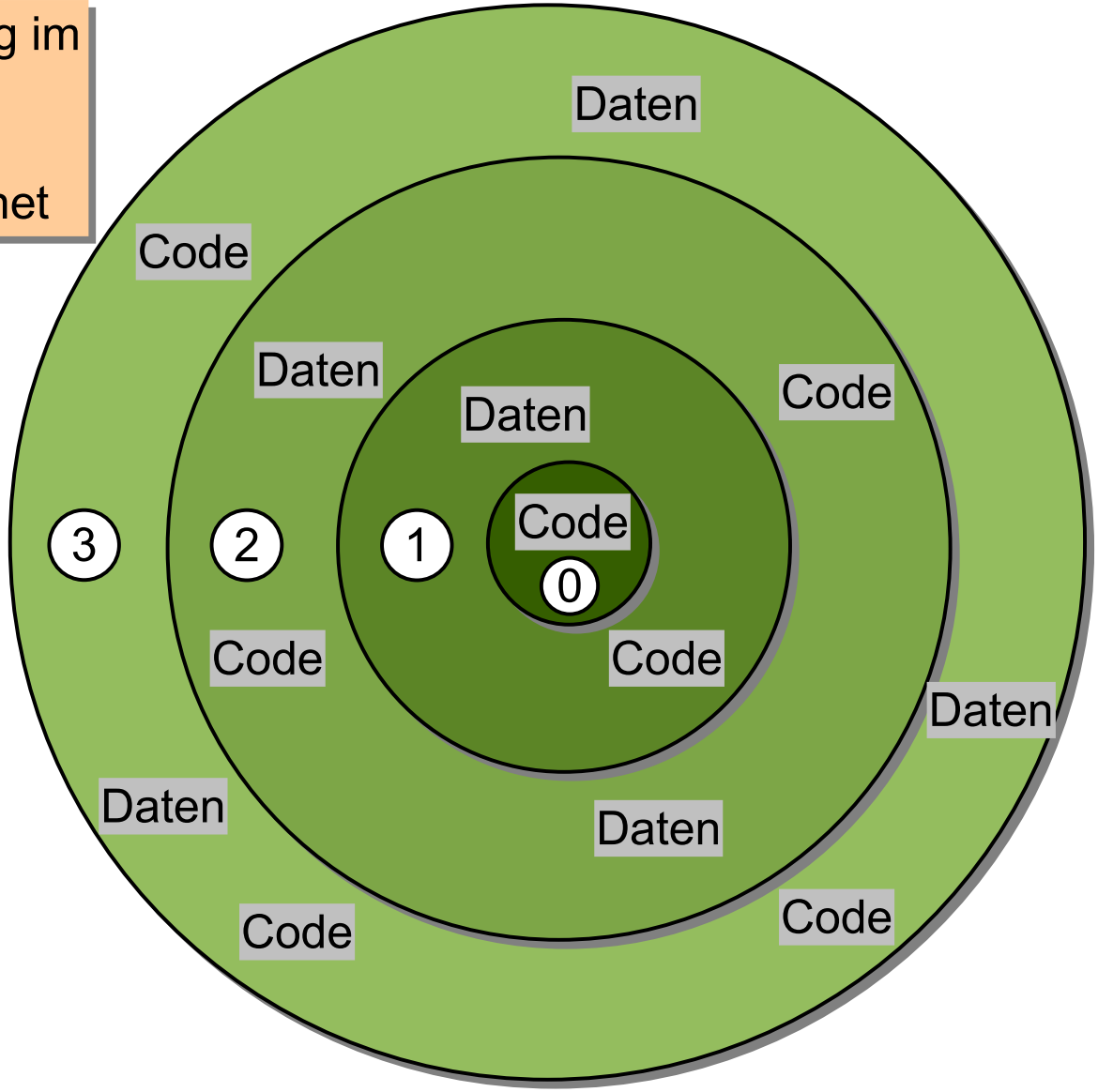
Schutz

TUHH
Technische Universität Hamburg

- 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")

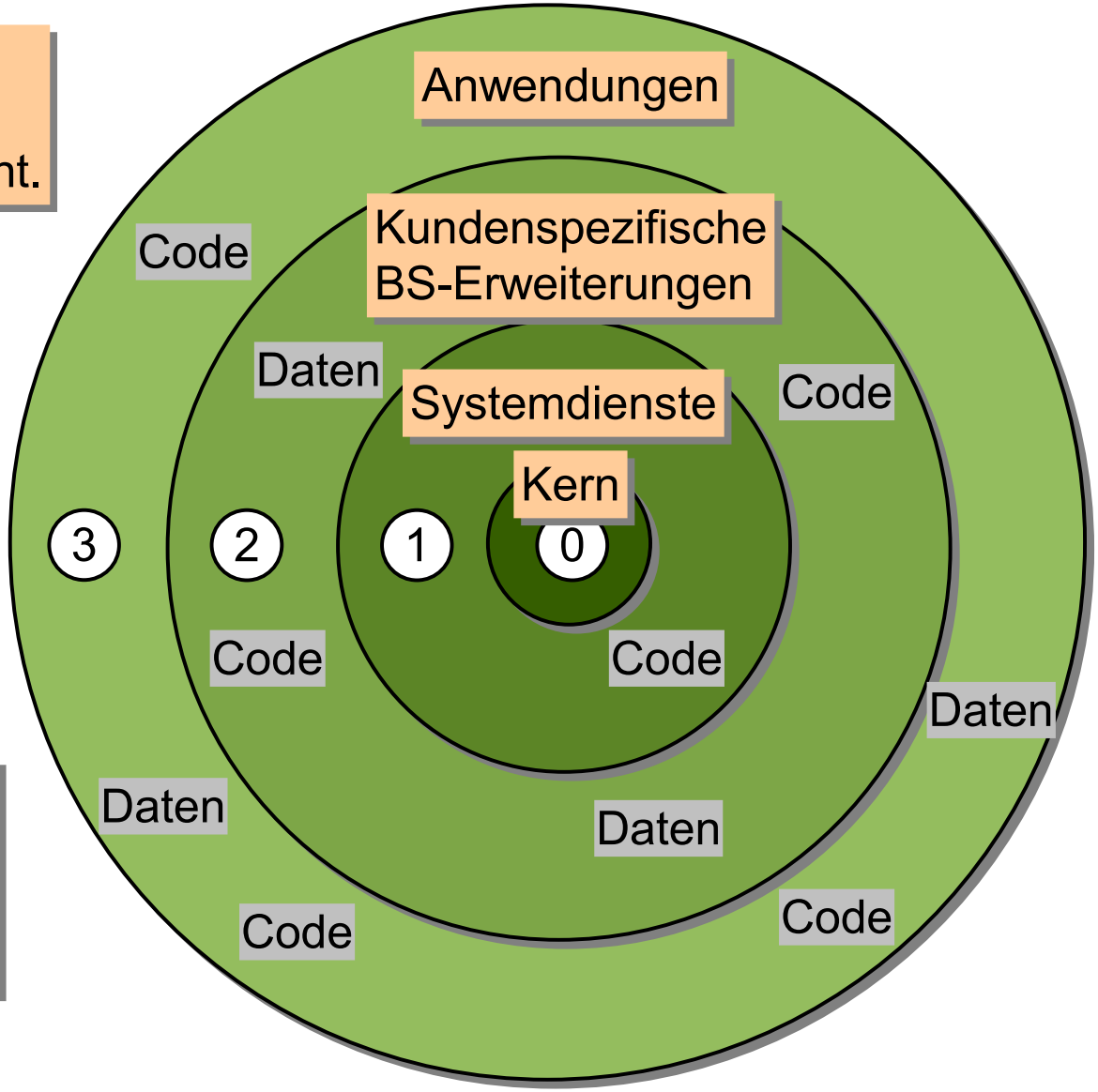
Schutzringe und *Gates*

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



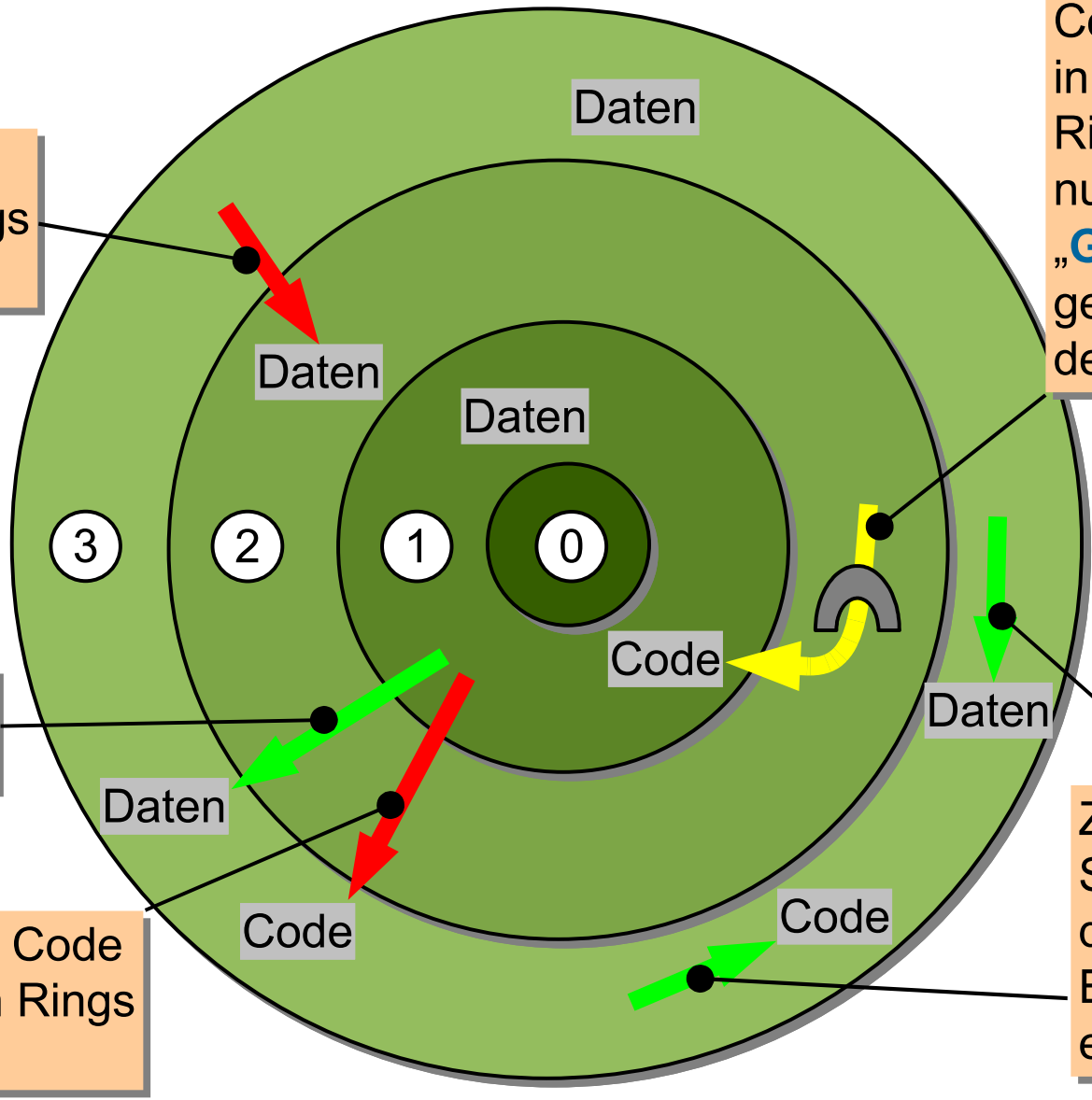
Schutzringe und *Gates*

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

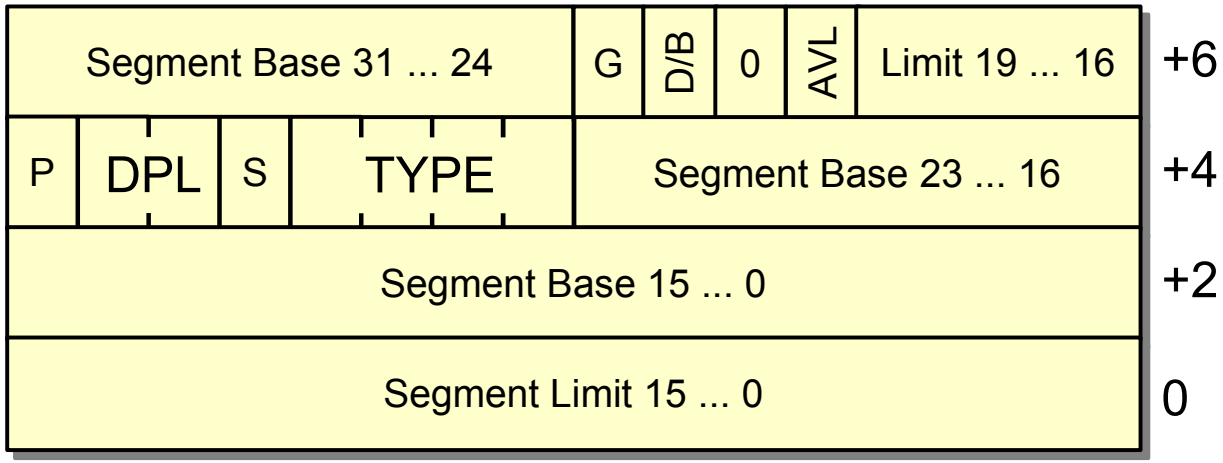
der Aufruf von Code eines äußeren Rings ist verboten!

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

TUHH
Technische Universität Hamburg

- 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)
```

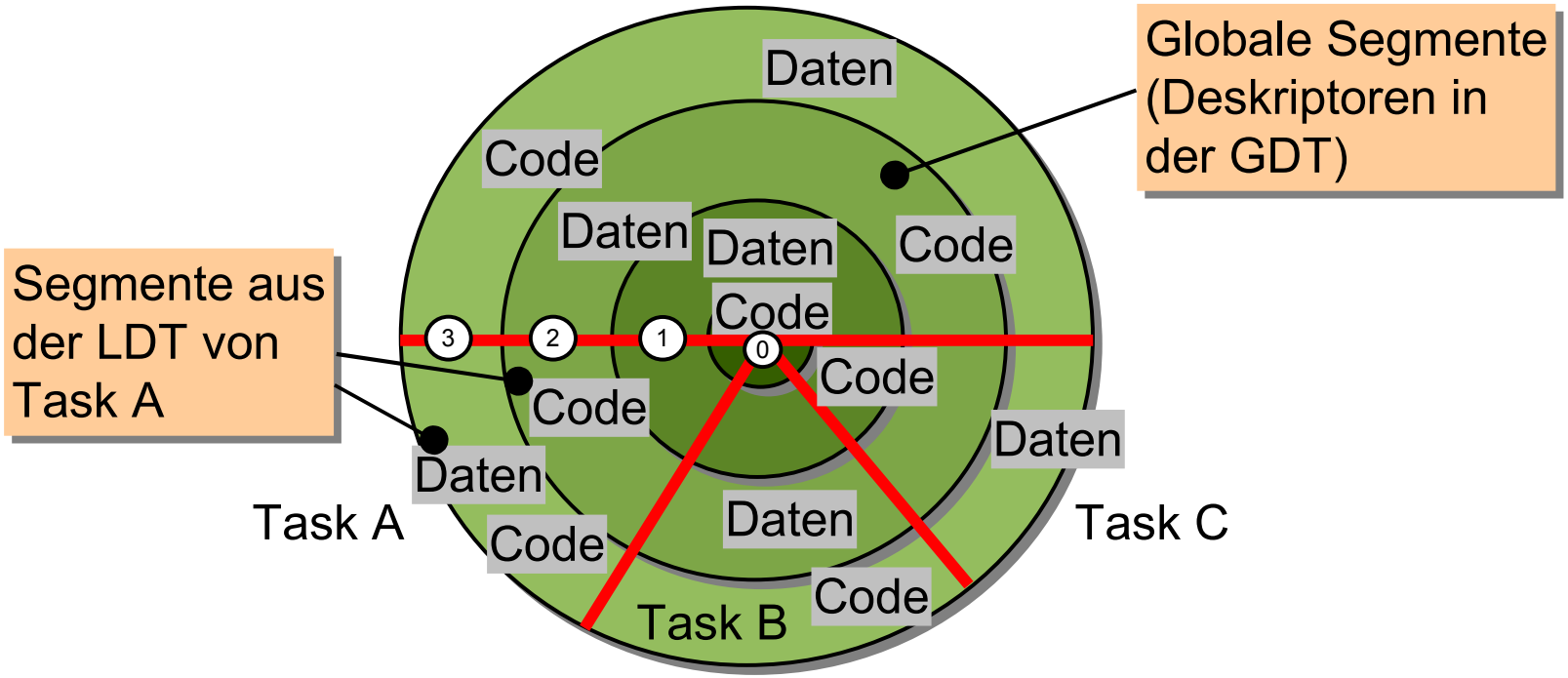
Agenda

TUHH
Technische Universität Hamburg

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

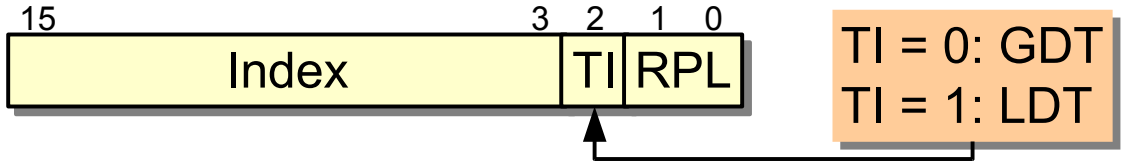
Multitasking

- 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 Segmentdeskriptortabelle

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



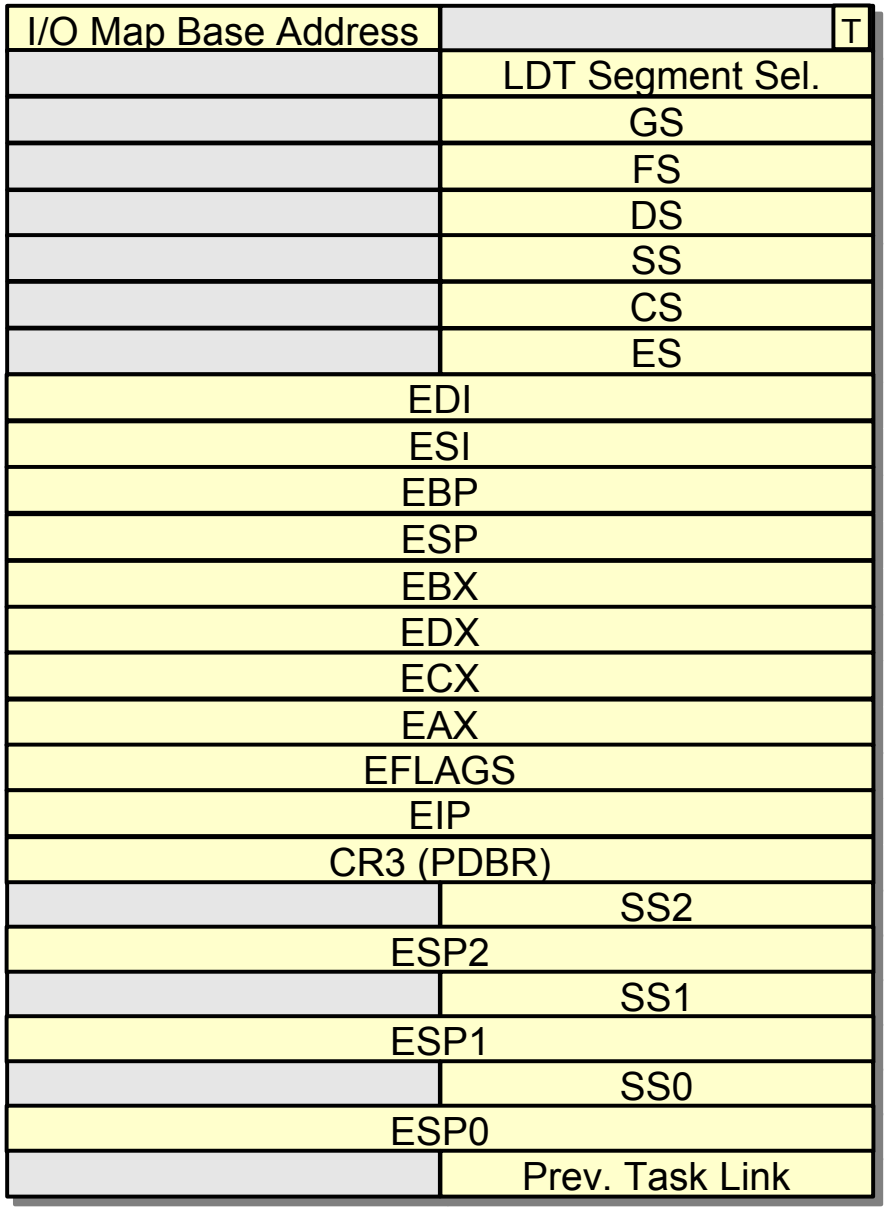
- ... werden mit Hilfe des **LDTR** gefunden, das bei jedem Taskwechsel ausgetauscht wird:

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!



Task Wechsel

TUHH
Technische Universität Hamburg

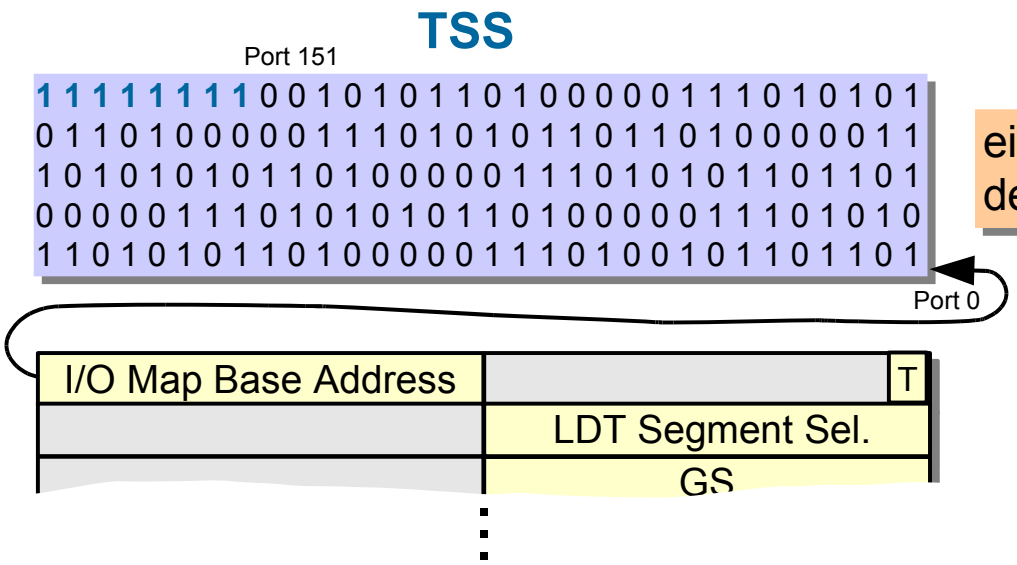
- für einen Task-Wechsel benötigt man entweder ...
  - ein Task-Gate in der GDT, einer LDT oder der IDT (Task-Wechsel bei Unterbrechungen!)
  - oder einfach nur einen TSS Deskriptor in der GDT
- ausgelöst werden kann ein Wechsel durch ...
  - eine JMP Instruktion
  - eine CALL Instruktion
  - eine Unterbrechung
  - 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:

Ports mit größeren Nummern dürfen nicht angesprochen werden

den Abschluss bildet immer ein Byte mit 0xff



eine '1' verhindert den Portzugriff

Agenda

TUHH
Technische Universität Hamburg

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

IA-32: Was gibt es sonst noch?

TUHH
Technische Universität Hamburg

- 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

TUHH
Technische Universität Hamburg

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

Zusammenfassung

TUHH
Technische Universität Hamburg

- 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"!

Nachwort: 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 im 64-Bit-Modus weggefallen!
  - Segmentierung (außer FS und GS Register)
    $\leadsto$ MMU *muss* benutzt werden
  - Task-Modell
  - Ring-Modell (außer -1, 0, 3)
  - Virtual-X86-Modus
- Und natürlich unterstützen auch alle AMD64-Prozessoren
  - IA32, A20-Gate, Real-Mode, ...