

# Betriebssystembau (BSB)

## VL 7 – IA-32: Das Programmiermodell der Intel-Architektur

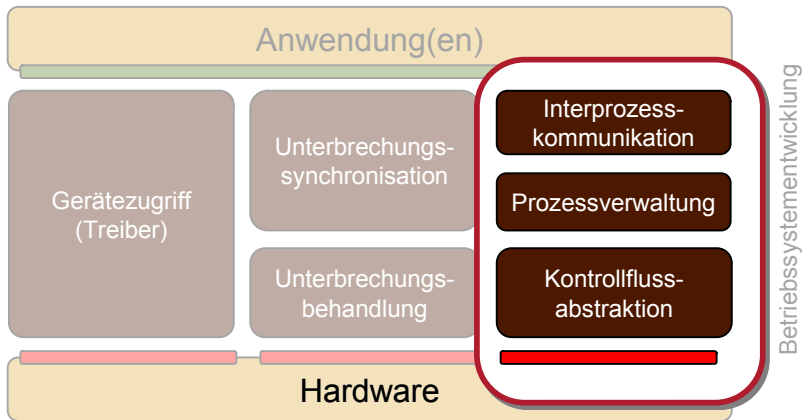
**Alexander Lochmann**

Lehrstuhl für Informatik 12 – Arbeitsgruppe Systemsoftware  
Technische Universität Dortmund

<https://sys.cs.tu-dortmund.de/de/lehre/ws23/bsb>

WS 23 – 13. November 2023

# Überblick: Einordnung dieser VL



# Agenda

---

Einordnung

Urvater: Der 8086

Die 32-Bit Intel-Architektur

Protected Mode

Multitasking

Zusammenfassung



- 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



- 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!



- 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*
  
- 2008: **Atom** *extrem stromsparend*
  - Architektur (wieder) CISC-lastiger, Ähnlichkeiten mit 486/Pentium
  
- 2009: **Core i7** *Nehalem-Architektur*  
*Sandy-Bridge-Architektur (2011)*  
*Haswell-Architektur (2013)*  
*Skylake-Architektur (2017)*
  - Hyperthreading, Octa Core, Quick Path Interconnect, AVX
  - "Power Control Unit" (PCU) passt Takt der TDP an
  
- 2012: **Xeon Phi** *Larrabee-Architektur*
  - P54C Manycore (62 cores), SIMD Instruktionen

Einordnung

Urvater: Der 8086

    Programmiermodell

    Speichermodell

Die 32-Bit Intel-Architektur

Protected Mode

Multitasking

Zusammenfassung



# 8086: Programmiermodell

- 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*

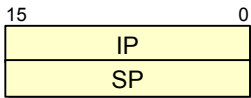
Aufwärtskompatibilität wird bei Intel groß geschrieben





# 8086: Registersatz

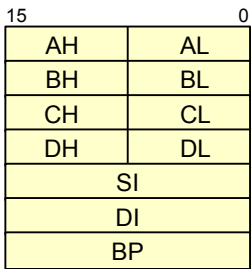
## Befehls- und Stapelzeiger



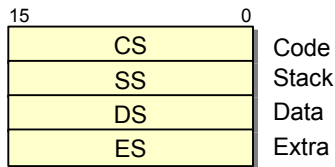
## Flag Register



## Vielzweckregister



## Segmentregister



# 8086: Registersatz

## Befehls- und Stapelzeiger

|    |   |
|----|---|
| 15 | 0 |
| IP |   |
| SP |   |

## Vielzweckregister

|    |    |
|----|----|
| 15 | 0  |
| AH | AL |
| BH | BL |
| CH | CL |
| DH | DL |
| SI |    |
| DI |    |
| BP |    |

### 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

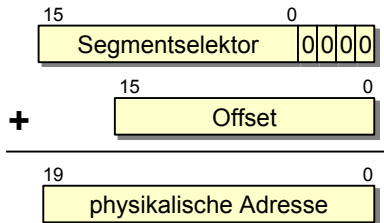
- für Array-Zugriffe (Displacement)

### BP: Base Pointer

Jedes „Vielzweckregister“ erfüllt seinen speziellen Zweck

# 8086: Segmentierter Speicher

- logische Adressen bestehen beim 8086 aus
  - Segmentsektor (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
  - Segmentsektor (i.d.R. der Inhalt eines Segmentregisters)
  - Offset (i.d.R. aus einem Vielzweckregister oder dem Befehl)
- Bei

*„640K ought to be enough for anybody“*

angeblich ein Zitat von Bill Gates, 1981

physikalische Adresse

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

# 8086: Speichermodelle

- 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



- 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



Einordnung

Urvater: Der 8086

**Die 32-Bit Intel-Architektur**

Erweiterungen

A20-Gate

Protected Mode

Multitasking

Zusammenfassung



# IA-32 – die 32 Bit Intel Architektur

- 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

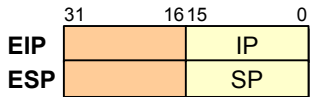




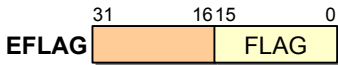
# 80386: Registersatz (Erweiterungen)

- erweiterte Register heißen aus Kompatibilitätsgründen E...

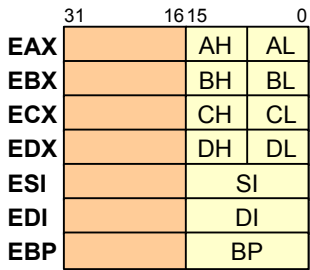
## Befehls- und Stapelzeiger



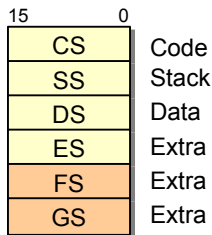
## Flag Register



## Vielzweckregister



## Segmentregister



Erweiterung  
zum 8086

# 80386: Registersatz (neue Register)

## 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 |   |

Erläuterungen  
folgen ...

## Steuerregister

|     | 31 | 16 15 | 0 |
|-----|----|-------|---|
| CR3 |    |       |   |
| CR2 |    |       |   |
| CR1 |    |       |   |
| CR0 |    |       |   |

## Testregister

|     | 31 | 16 15 | 0 |
|-----|----|-------|---|
| TR7 |    |       |   |
| TR6 |    |       |   |

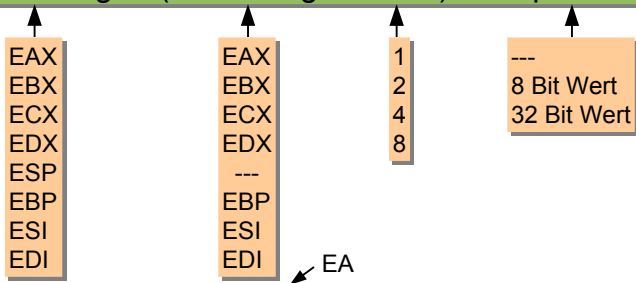
## Debugregister

|  | 31 | 16 15 | 0 |
|--|----|-------|---|
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |
|  |    |       |   |

# IA-32: Adressierungsarten

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

$$\text{EA} = \text{Basis-Reg.} + (\text{Index-Reg.} * \text{Scale}) + \text{Displacement}$$



- 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:

$$\begin{array}{r} 0\text{x}\text{ffff}0 \leftarrow \text{Segment} * 16 \\ + 0\text{x}0\text{fff}f \leftarrow \text{Offset} \\ \hline \text{0x0ffef} \leftarrow \text{nur 20 Bit!} \quad \text{0x10ffef} \leftarrow \text{phys. Adresse} \end{array}$$

- 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 adressieren
- Ab dem 486 hat Intel das A20-Gate in die CPU integriert!



# Agenda

---

Einordnung

Urvater: Der 8086

Die 32-Bit Intel-Architektur

**Protected Mode**

    Segmente

    MMU

    Schutz

Multitasking

Zusammenfassung

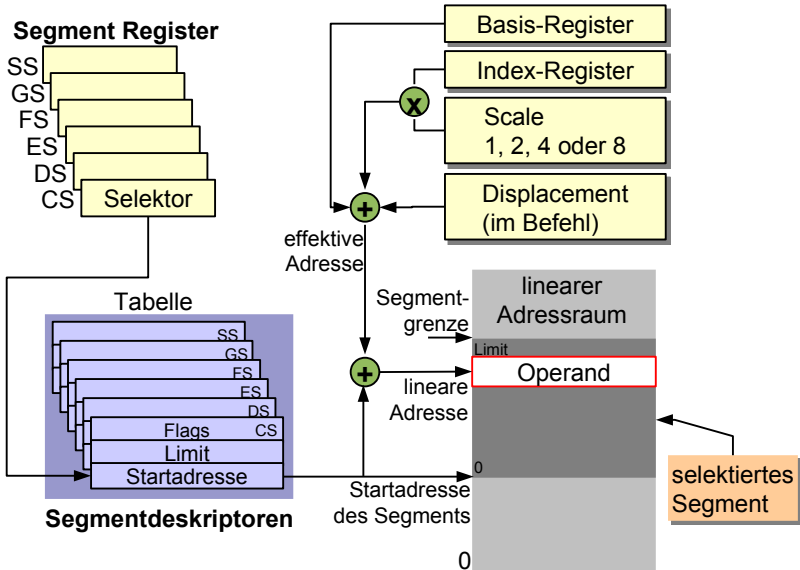


# IA-32: Protected Mode – Segmente

- 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.

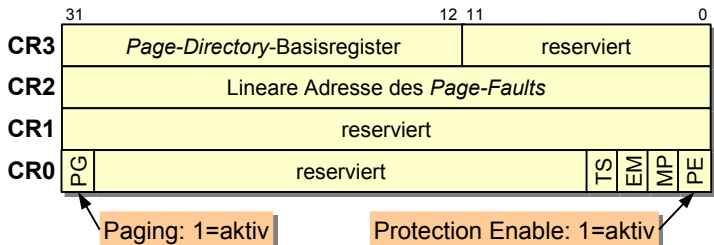


# IA-32: Protected Mode – Segmente



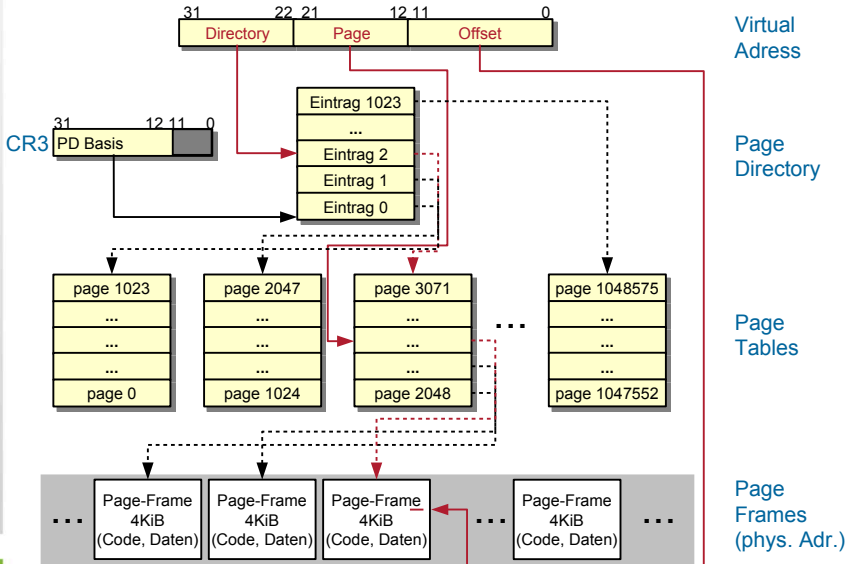
# IA-32: Seitenbasierte MMU (1)

- 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)



- 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

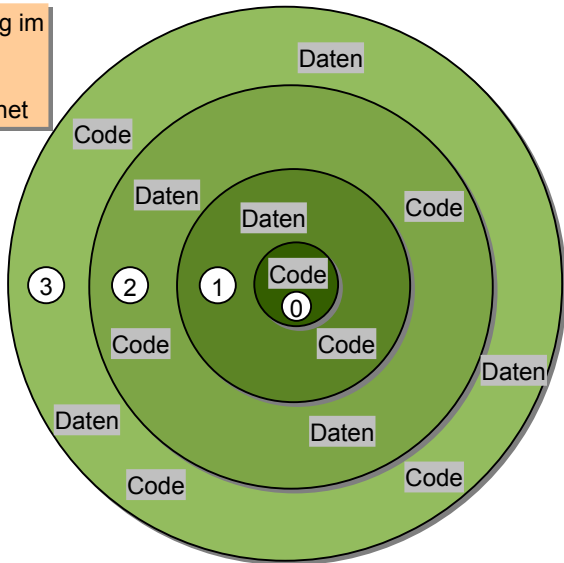


- die wichtigste Eigenschaft des IA-32 Protected Mode ist das Schutzkonzept
- **Ziel:** fehlerhaften oder nicht vertrauenswürdigen Code isolieren
  - Schutz vor Systemabstürzen
  - Schutz vor unberechtigten Datenzugriffen
  - keine unberechtigten Operationen, z.B. I/O Port Zugriffe
- Voraussetzungen: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



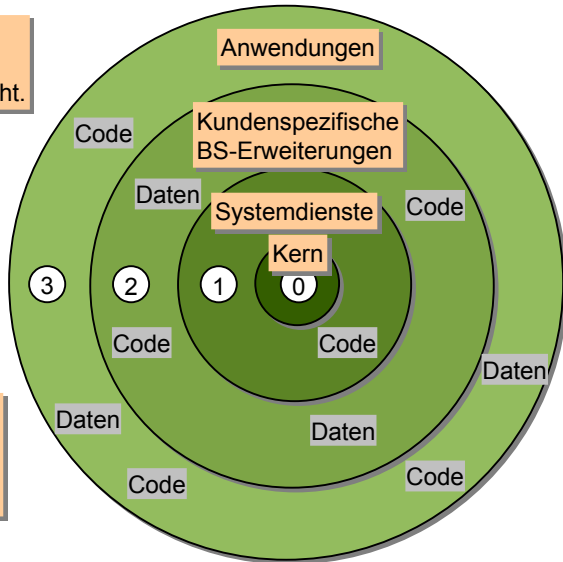
# Schutzringe und Gates

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



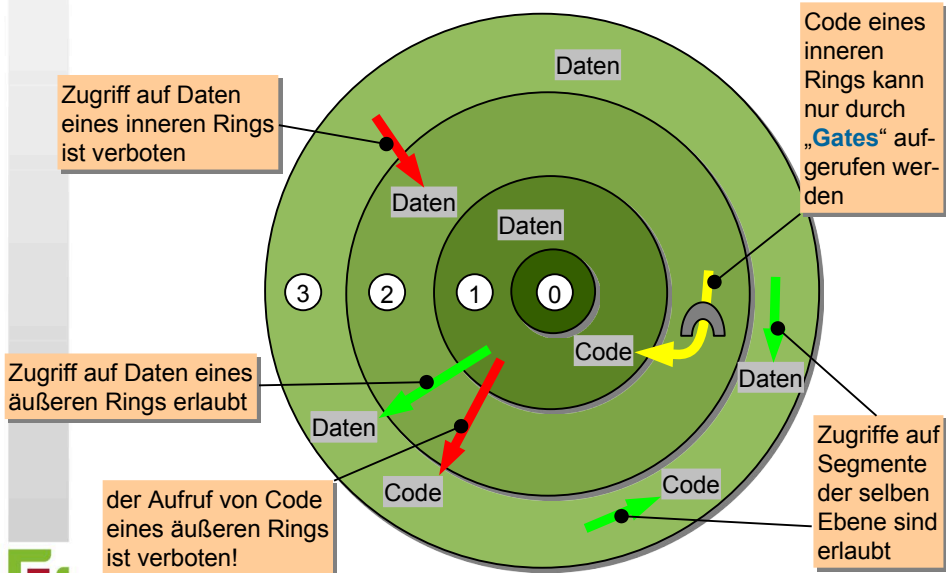
# Schutzringe und Gates

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



Privileg-Ebene 0 ist die höchste und dem Betriebssystemkern vorbehalten.

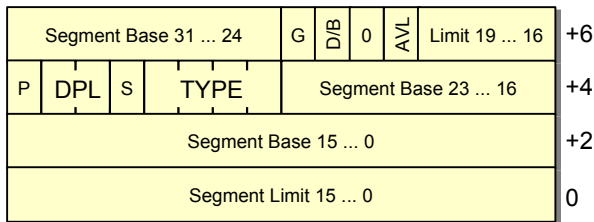
# Schutzringe und Gates



# 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

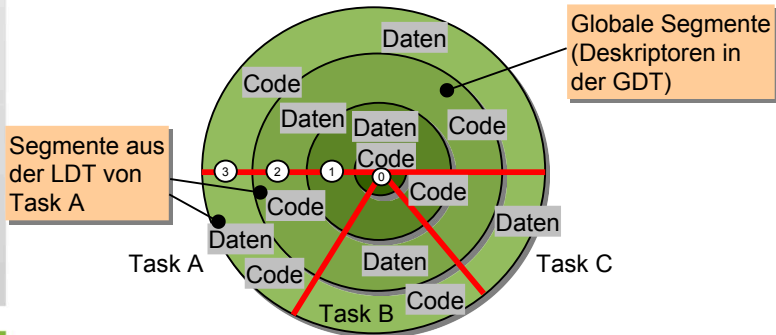
**Multitasking**

Zusammenfassung



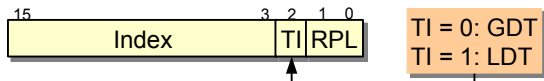
# Multitasking

- 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 |   |
| <b>LDTR</b> | 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      |     |

- 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.

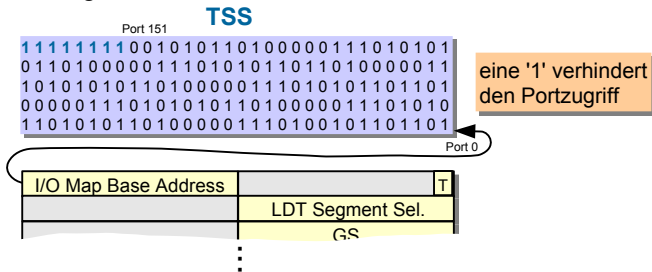


# Ein-/Ausgaben im Protected Mode

- 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



# 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 IA-32 Architektur ist ausgesprochen komplex
  - Was ist mit 64-Bit?  $\rightsquigarrow$  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)
      - $\rightsquigarrow$  MMU *muss* benutzt werden
    - Task-Modell
    - Ring-Modell (außer -1, 0, 3)
    - Virtual-X86-Modus
- Stichwort "PIC" und "A20 Gate"! 😊