

□ **Anmeldung zur TI-Klausur:**

- Einwurf der Zulassungsbescheinigung in den Briefkasten im Untergeschoss des Informatikgebäudes am Fasanengarten **bis spätestens 02. September**. Es handelt sich um den gleichen Briefkasten, in dem die Übungsblätter eingeworfen werden **UND**
- **Online-Anmeldung auf der TI-Homepage**

□ Hilfsmittel sind nicht erlaubt

□ Dauer der Klausur:

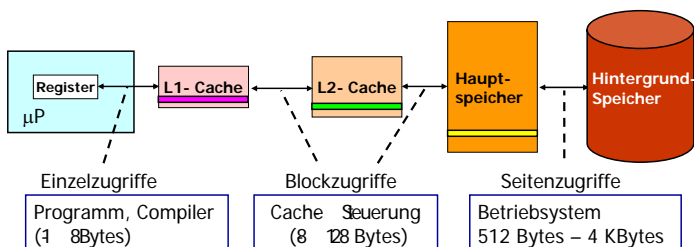
- **Informatik: 120 Minuten** (14.00 - 16.00 Uhr)
- **Informationswirtschaft: 60 Minuten** (14.00 - 15.00 Uhr)

□ Studentenausweise unbedingt in die Klausur mitbringen

□ Hörsaalverteilung wird rechtzeitig bekannt gegeben (TI Homepage)

## Speicherhierarchie

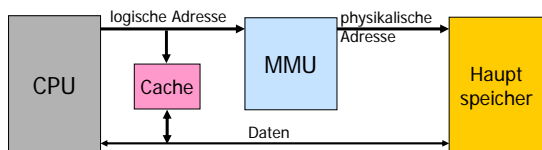
Daten werden nur zwischen aufeinanderfolgenden Ebenen der Speicherhierarchie kopiert.



## Cache und Speicherverwaltungseinheit

Zwei Möglichkeiten der Cache-Einbindung bei virtueller Speicherverwaltung:

- **Virtueller Cache:** wird zwischen CPU und MMU gelegt. Die höherwertigen Bits der logischen Adressen als Tags abgelegt



## Virtueller und physikalischer Cache

□ **Vorteile des virtuellen Caches:**

- bei Treffern wird die MMU nicht benötigt → Keine Verzögerung durch die Adressberechnung der MMU

□ **Vorteile des physikalischen Caches:**

- physikalische Adresse ist i. A. viel kleiner als die logische Adresse → weniger Bits müssen als Tag gespeichert werden.
- befindet sich die MMU auf dem Prozessorchip, so kann nur der physikalische Cache außerhalb erweitert werden.

# Virtuelle Speicherverwaltung

Software-Prinzipien und Hardware-Unterstützung der virtuellen Speicherverwaltung

## Segmentierung vs. Seitenwechsel

□ **Segmentierung (Segmente variabler Größe)**

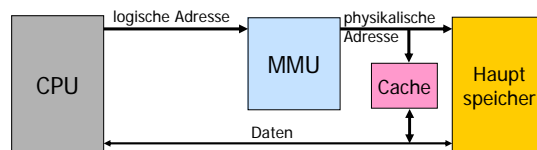
- logische Abbildung einer Programmstruktur
- geringer Datentransfer
- umfangreicher Datentransfer beim Ein-/Auslagern
- externe Fragmentierung

□ **Seitenwechsel-Verfahren (Seiten fester Größe)**

- geringerer Verwaltungsaufwand
- bessere Hauptspeicherauslastung
- häufiger Datentransfer
- interne Fragmentierung

## Cache und Speicherverwaltungseinheit

- **Physikalischer Cache:** wird zwischen MMU und Speicher gelegt. Die höherwertigen Bits der physikalischen Adressen werden als Tags abgelegt



## Segmentorientierte Speicherverwaltung

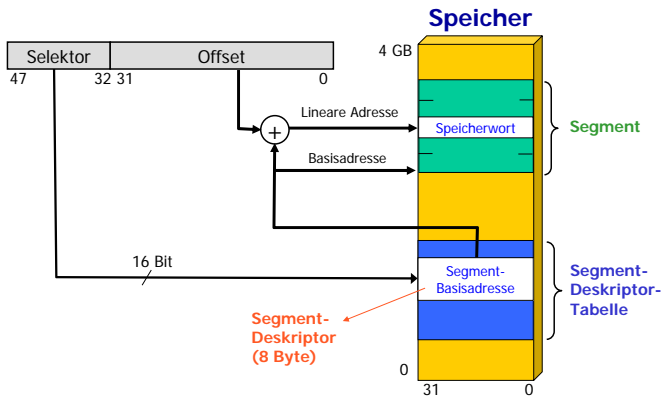
Der Speicher wird in ein oder mehrere physikalische Segmente (zusammenhängende Speicherbereiche variabler Länge) unterteilt

Oft verschiedene Segmenttypen (Code- und Datensegmente)

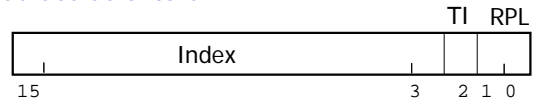
Maximale Segmentlängen je nach Prozessor zwischen 64 KByte und bis 4 GByte

**Fallbeispiel x86-Prozessoren:**

Berechnung der physikalischen aus der logischen Adresse



Aufbau des Selektors:



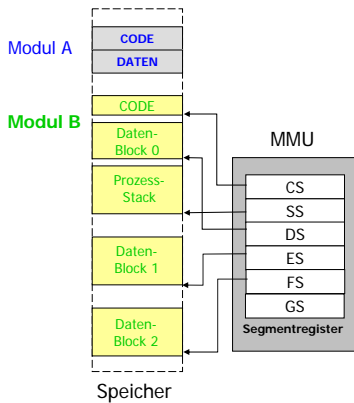
- Indexfeld und TI (Tabellenindikator) selektieren einen Eintrag in der Segment-Deskriptor-Tabelle
- RPL-Bits (Requested Privilege Level) geben eine Privileg-Ebene an, die ein Befehl besitzen muss, um auf das gewünschte Segment zugreifen zu dürfen.

**Segment-Deskriptoren:** Jeder Segment-Deskriptor beschreibt das zugehörige Segment durch folgende Attribute:

- Segment-Basisadresse (*base address*)
- Segment-Größe in Bytes (*limit*)
- Zugriffsrechte auf das Segment (*access rights*) zur Realisierung von Schutzmechanismen.

- Jedes Speicherwort wird durch einen **Segment-Selektor**, der den Segmentanfang kennzeichnet, und einen **Offset** innerhalb des Segments adressiert
- Durch Lokaltätseigenschaften in Programmen wird nicht bei jedem Zugriff auf den Hauptspeicher ein neues Segment benutzt → Segment Selektoren wechseln selten.
- Aus Geschwindigkeitsgründen werden die Segment Selektoren in speziellen Segment Registern gespeichert. Hierzu existieren verschiedene Register für verschieden Segment Typen: Code Segment (CS), Stack Segment (SS), Daten Segment (DS), Extra Segment (ES, FS, GS, ...) → Adressierung eines Speicherworts durch ein **Segment-Register** und ein **Offset**
- Wichtige Informationen über das ausgewählte Segment werden in einem **Segment-Deskriptor-Cache** abgelegt → keine Speicherzugriffe beim Lesen der Segmenteigenschaften

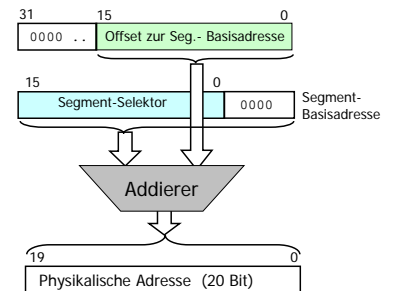
- Jedes Segment, dessen Anfangsadresse in einem Segmentregister abgelegt ist, befindet sich physikalisch im Hauptspeicher.
- Die durch die Segmentregister spezifizierten Segmente bilden die Arbeitsmenge (*working set*) eines Prozesses



Adressierungs-Modi (x86-Prozessoren)

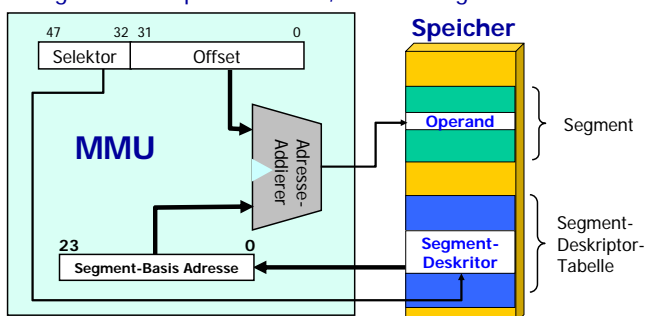
Um Kompatibilität zu den älteren Mikroprozessoren innerhalb des x86 Prozessorfamilie zu gewährleisten, können die Prozessoren in in verschiedenen Adressierungsmodi arbeiten:

- Real (Address) Mode** (Kompatibilitätsmode zu 8086)
  - 20 Bit lange physikalische Adressen
  - 1 MByte max. adressierbare Hauptspeicherkapazität
  - max. Segmentlänge: 64 KByte



Adressierungs-Modi (x86-Prozessoren)

- Protected (Virtual Address) Mode** Modus mit vergrößertem Adressraum und erweiterten Fähigkeiten zu Speicherschutz, Multitasking



Adressierungs-Modi (x86-Prozessoren)

- Abbildung eines virtuellen Adressraums von mehreren Tera-Bytes auf einen 4 GByte großen physikalischen Adressraum
- Der Segment-Selektor spezifiziert hier nicht die Basisadresse des Segments selbst, sondern verweist auf den **Segment-Deskriptor** in der **Segment-Deskriptor-Tabelle** im Hauptspeicher
- Jeder Segment-Deskriptor beschreibt das zugehörige Segment durch folgende Attribute:
  - die Segment- Basisadresse (*base address*)
  - die Segment- Größe in Bytes (*limit*)
  - die Zugriffsrechte auf das Segment (*access rights*)

# Seitenorientierte Speicherverwaltung

Der Speicher wird in viele kleine Seiten gleicher Länge unterteilt

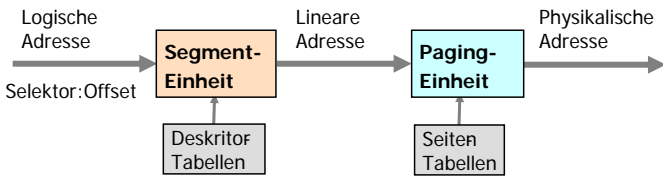
Seitenlängen von 256 Byte bis 8 kByte

Im Unterschied zu Segmenten können Seiten nicht an beliebiger Stelle im Speicher beginnen, sondern nur in einem festen an der Seitengröße orientierten Raster

## Fallbeispiel x86-Prozessoren (ab 80386):

Aus Kompatibilitätsgründen zum 80286 unterstützen die x86-Prozessoren ab dem 80386 sowohl eine segment- als auch eine seitenorientierte Speicherverwaltung

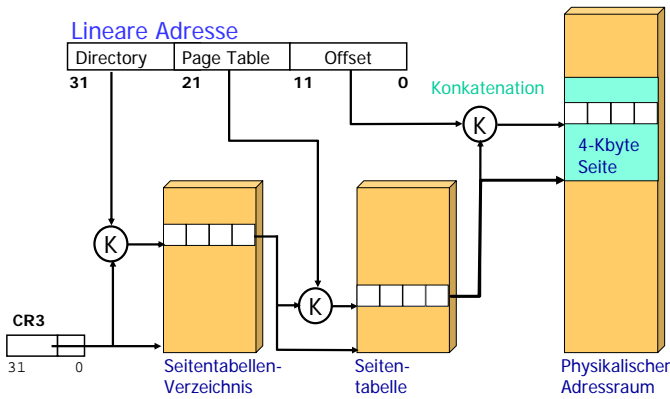
## Berechnung der physikalischen Adressen (x86-Prozessoren)



Zunächst wird wie beim 80286 die logische Adresse über eine Segment-Verwaltung umgesetzt. Die entstehende Adresse heißt hier **lineare Adresse**

Im Unterschied zum 80286 wurde die logische Adresse beim 80386 jedoch auf 48 Bit erweitert, die lineare Adresse (entspricht der physikalischen beim 80286) auf 32 Bit

## Zweistufiges Seitenwechsel-Verfahren

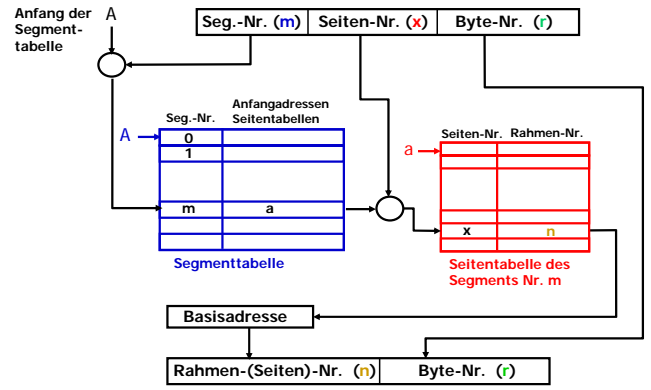


## Anmerkungen

Sowohl bei segmentorientierter wie bei seitenorientierter Speicherverwaltung gilt:

Befindet sich eine Seite oder ein Segment nicht im Hauptspeicher, so löst der Prozessor eine Unterbrechung aus, um die Seite oder das Segment durch das Betriebssystem zu laden (Seiten- oder Segmentfehler).

# Segmentierung mit Seitenwechsel



## Berechnung der physikalischen Adressen (x86-Prozessoren)

Die nachgeschaltete Seiten-Verwaltung kann optional abgeschaltet werden

→ Kompatibilitätsmode zum 80286, lineare Adresse = physikalische Adresse

Anderenfalls wird aus der linearen Adresse durch die Seiten-Verwaltung die physikalische Adresse ermittelt. (bei Prozessoren mit ausschliesslich Seiten-Verwaltung entspricht die lineare Adresse somit der logischen Adresse)

## Zweistufiges Verfahren zur Berechnung phys. Adressen:

## Zweistufiges Seitenwechsel-Verfahren

- Seitentabellenverzeichnis → Seitentabellen → Seiten
- Seitentabellenverzeichnis, Seitentabellen und Seiten sind jeweils 4 KByte groß und jeder Tabellen Eintrag umfasst 4 Byte
- In den Seitentabellen- und -verzeichnis sind jeweils 1024 Einträge enthalten.
- Lineare Adresse wird in drei Teilen zerlegt. Die höchstwertigen 10 Bits selektieren einen Eintrag im Seitentabellen -verzeichnis, dessen Basisadresse in einem speziellen Systemregister (CR3) abgelegt ist.
- Die nächsten 10 Bits der linearen Adresse selektieren einen der 1024 Einträge aus der Seitentabelle. In diesem Eintrag steht die Basisadresse einer Seite.
- Die niedrigstwertigen 12 Bits werden als Offset zur Seitenadresse addiert, um die endgültige, physikalische Adresse zu erhalten.

## Anmerkungen

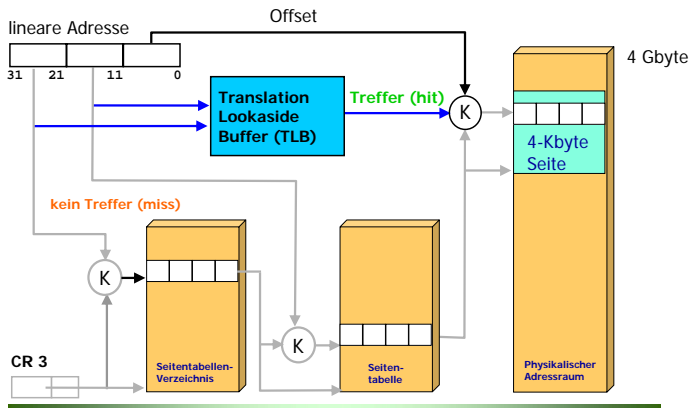
### Erkennung eines Segmentfehlers:

Bit im Segment-Deskriptor zeigt an, ob das Segment im Hauptspeicher ist oder nicht.

### Erkennung eines Seitenfehlers:

Seite oder Seitentabelle befindet sich nicht im Hauptspeicher (Seitenfehler).

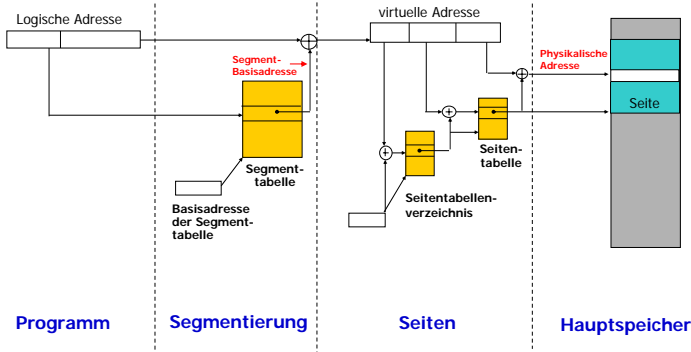
Spezielles Kennungbit im Seitentabellen-Verzeichnis (Seitentabellenfehler)



Ein schneller voll-assoziativer Cache (*Translation Lookaside Buffer, TBL*) speichert automatisch die zuletzt benutzten Einträge aus dem Seitentabellenverzeichnis und der Seitentabelle

➔ Im Trefferfall (Trefferquote ca. 90 %) muss nicht auf die im Hauptspeicher liegenden Seitentabellen zugegriffen werden

### Automatische Adressübersetzung bei 80486 und Pentium-Prozessoren



### Schutzmechanismen

Moderne Mikroprozessoren bieten Schutzmechanismen an, um während der Laufzeit von Programmen unerlaubte Speicherzugriffe zu verhindern. Dies geschieht im wesentlichen durch:

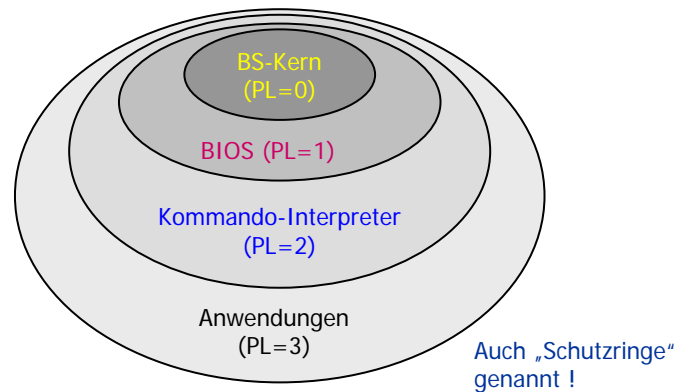
- Trennung der Systemsoftware, z. B. des Betriebssystems, insbesondere des Ein-/Ausgabe-Subsystem (BIOS, basic I/O system), von den Anwendungsprozessen.
- Trennung der Anwendungsprozessen voneinander. Ist dies nicht gewährleistet, könnte ein fehlerhaftes Anwenderprogramm andere, fehlerfreie Programme beeinflussen (Schutzebenen und Zugriffsrechte)

### Schutzmechanismen

**Schutzebenen (PL: Privilege Levels):** Wichtigstes Mittel zur Realisierung von Schutzmechanismen

- Zweischieutzebenen (bei Seitenverwaltung der x86-Prozessoren):
  - Betriebs Systemmodus (supervisor mode)
  - Benutzermodus (user mode)
  - Ein Auftrag im Benutzermodus darf keine Daten und Programme des höherprivilegierten Betriebssystemmodus benutzen.
- Vierstufige Hierarchie bei Segmentverwaltung:
  - Privileg Ebene PL = 0 entspricht der vertrauenswürdigsten Ebene (most trust level). Privileg Ebene PL = 3 entspricht der am wenigsten vertrauenswürdigsten Ebene (least trust level)

### Beispiel eines Systems mit vier Schutzebenen



### Regeln für den Zugriffsschutz (*protection rules*)

- Ein Prozess darf nur auf Daten zugreifen, die höchstens genauso vertrauenswürdig (*trusted*) sind wie er selbst
- Ein Prozess darf nur Code benutzen, der mindestens genauso vertrauenswürdig ist wie er selbst
- Zugriffsrechte (*access rights*) garantieren, dass nur unter bestimmten Voraussetzungen auf die im Speicher abgelegten Informationen zugegriffen werden darf.

Sowohl die Schutzebenen als auch die Zugriffsrechte werden hardwaremäßig bei der Speicherverwaltung durch die Vergabe von Privileg-Ebenen und Rechten an Speichersegmente und Speicherseiten unterstützt.

### Kapitel 9

- Zeitverhalten des Bussystems
- Systemsteuer- und Schnittstellenbausteine
- Ausnahmebehandlung

- Synchroner Systembus
- Semi-Synchroner Systembus
- Asynchroner Systembus

## Timing

Adresse wird zu Beginn des Buszyklus ( $T_a$ ) auf den Adressbus gelegt  
Auswahl der Übertragungsrichtung durch  $R/\overline{W}$

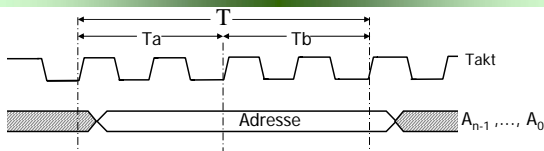
### Lesen ( $R/\overline{W} = 1$ ):

- Speicher (oder andere Systemkomponente) liefert ihre Daten gegen Ende des Buszyklus ( $T_b$ )
- Übernahme der Daten in den Prozessor mit der steigenden Flanke des Systemtakts

### Schreiben ( $R/\overline{W} = 0$ ):

- Prozessor legt die Daten zu Beginn der zweiten Taktthälfte ( $T_b$ ) auf den Systembus
- Übernahme der Daten in den Speicher durch die steigende Flanke von  $R/\overline{W}$  oder des Systemtakts

## Semi-synchroner Systembus



## Semi-synchroner Systembus

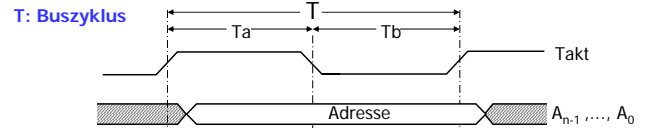
Um auch langsamere Speicher benutzen zu können:

### Steuereingang $\overline{READY}$

Buszyklus wird nur abgeschlossen, wenn rechtzeitig vor Ende von der Taktthälfte ( $T_b$ )  $\overline{READY} = 0$  ist

Ist  $\overline{READY}$  am Ende des Buszyklus nicht 0

- ➔ Es werden solange **Wartezyklen** ( $T_w$ , Dauer: z. B. halbe Buszykluslänge) eingefügt, bis  $\overline{READY} = 0$  wird



## Synchroner Systembus

- Alle Vorgänge synchron zum Takt nach einem starren Muster ablaufen ➔ **synchroner Systembus**
- Übergabe und Übernahme der Daten geschieht zu festgelegten Taktflanken
- Synchroner Busse in den Anfangsjahren der  $\mu$ Pen
- **Nachteil:** Alle am Bus angeschlossenen Komponenten müssen strenge Zeitvorgaben erfüllen
  - ➔ Langsamere Komponente bestimmt die zulässige Geschwindigkeit des Busses, oder aber der Bus schließt den Einsatz von „schnellen“ Komponenten aus (z. B. keine schnelle Speicher ☹)

## Timing

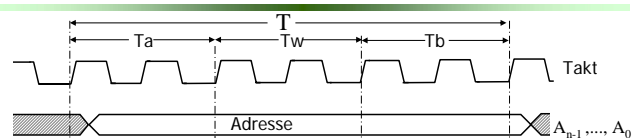
### Lesen ( $R/\overline{W} = 1$ ):

- hinreichend schneller Speicher liefert seine Daten im letzten Taktzyklus von ( $T_b$ )
- Übernahme der Daten in den Prozessor durch die steigende Taktflanke des nächsten Buszyklus

### Schreiben ( $R/\overline{W} = 0$ ):

- Prozessor legt die Daten zu Beginn der zweiten Taktthälfte von ( $T_b$ ) auf den Systembus
- Übernahme der Daten in den Speicher durch die steigende Flanke von  $R/\overline{W}$

## Einfügen eines Wartezyklus



## Semi-synchroner Systembus

- Moderne Prozessoren besitzen höhere Taktfrequenzen
- Schreibe- bzw. Lesezugriffe benötigt mehrere Taktzyklen
- **Steuereingänge (READY)** zur Synchronisation der Buszugriffe durch Einführung von Wartezyklen (*wait states*) → unterschiedlich schnelle Speicher und Geräte können individuell bedient werden
- Bezeichnung: Semi-synchrone Busse (Synchrone Busse mit Wartezyklen)
- **Timing:**
  - Adresse zu Beginn des Buszyklus ( $T_a$ ) auf den Adressbus
  - Auswahl der Übertragungsrichtung wieder durch  $R/\overline{W}$



## Semi-synchroner Systembus

- Bus ist immer noch synchron (streng am Takt orientiert), die Dauer eines Buszyklus ist jedoch nicht mehr fest, sondern in Vielfachen von Taktzyklen variierbar
  - ➔ **semi-synchroner Systembus**
- höherer Steueraufwand als synchroner Systembus
- Zeitverhalten ist auf verschieden schnelle Bausteine anpassbar
- Sind nur ausreichend schnelle Bausteine im System, kann READY z. B. fest auf 0 gelegt werden. Anderenfalls kann dieses Signal durch eine geeignete Verzögerungsschaltung (z. B. Monoflop) erzeugt werden

