

# Multi-Tasking

*Mehrere Prozesse sollen (fast) gleichzeitig auf einem Rechner ablaufen.*

*Prozessabläufe sollen voneinander unabhängig sein:*

- CPU wird von den Prozessen abwechselnd genutzt.
- Prozesse nutzen verschiedene I/O-Geräte.
- Prozesse nutzen verschiedene Speicherbereiche.

*Prozessabläufe sollen koordinierbar sein:*

- Prozesse nutzen gemeinsame Speicherbereiche.

# Multi-Tasking

*Prozesse*

- können freiwillig kooperieren

*oder*

- Hardware kann dies durch nachfolgende Erweiterungen erzwingen

*Nutzung der CPU:*

Hardware verhindert zu lange Nutzung der CPU durch einen Prozess.

*I/O-Koordinierung:*

Hardware verhindert direkten Zugriff.

*Speicherschutz:*

Hardware verhindert Lese-/Schreibzugriff auf „fremde“ Speicherbereiche.

# Multi-Tasking

## *Probleme (Beispiele):*

- Prozesse müssen auf I/O-Geräte zugreifen können.
- Beim Starten oder Terminieren von Prozessen muss auf fremde Speicherbereiche zugegriffen werden.
- Beim Allozieren neuen Speichers muss der Speicherschutz geändert werden.
- ...

## *Je nach Sicherheitsstrategie ist manches erlaubt, manches nicht. Beispiele:*

- Wenn der Prozess vorher eine Datei geöffnet hat, dann darf er im weiteren Verlauf auf die zugehörigen Blöcke der Platte zugreifen.
- Wenn der Prozess vorher Speicher alloziert hat, dann darf er auf den Speicher zugreifen.
- ...

## *Sicherheitsstrategie ist ein Programm!*

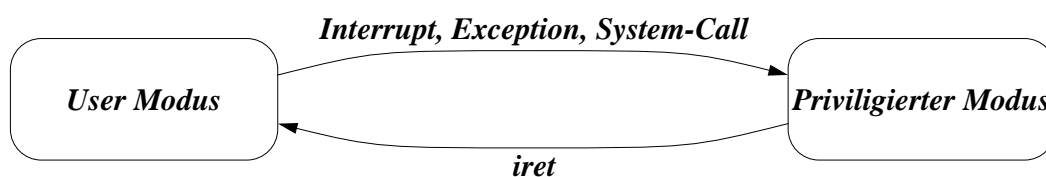
# Multi-Tasking

## *Dieses „Strategie-Programm“ (Betriebssystem) muss den laufenden Prozessen*

- Rechte geben bzw. entziehen können
- oder für die Prozesse Aktionen durchführen.

## *Lösung:*

*Die CPU führt bestimmte Instruktionen nur dann aus, wenn sie in einem bestimmten Zustand ist (Priviligierungs-Stufen). Zustands- / Priviligierungswechsel z.B. nur über Interrupts, Exceptions und System-Calls sowie „iret“.*



# Multi-Tasking

*Interrupt-, Exception- und System-Call-Bearbeitung heißt also:*

- Abspeichern des Instruction-Pointers
- Abspeichern einiger Register  
(z.B. Condition-Code- und Interrupt-Enable-Register)
- Abspeichern des CPL (Current Privilege Level)

*sowie*

- Laden des neuen Instruction-Pointers
- Laden neuer Register-Werte  
(z.B. neuer Wert für das Interrupt-Enable-Register)
- Laden des neuen CPL

# Multi-Tasking

*Da Interrupt-, Exception- und System-Call-Handler jetzt privilegiert ablaufen, dürfen die Interrupt-, Exception- und System-Call-Vektoren im User-Modus nicht modifiziert werden können.*

*Dies kann erreicht werden, indem die Interrupt-, Exception- und System-Call-Vektor-Tabelle im Speicher liegt und dieser im User-Modus nicht zugänglich ist (ähnlich den Speicherbereichen anderer Prozesse).*

# Multi-Tasking

*CPU wird abwechselnd von allen Prozessen verwendet. Prinzip:*

```
forever {
    Load registers of process #1
    Do some work...
    Save registers of process #1

    Load registers of process #2
    Do some work...
    Save registers of process #2

    ...

    Load registers of process #N
    Do some work...
    Save registers of process #N
}
```

**Problem:** „Do some work...“ *muss immer wieder terminieren!*

# Multi-Tasking

*Unterbrechen / Abbrechen eines Programm-Stückes über (Timer-) Interrupts möglich.*

**Timer-Interrupt-Handler:**

```
timer_handler:
    /* Save registers of current process. */
    pushl %eax; pushl %ebx; ...; pushl %esi
    movl current, %eax
    movl %esp, (%eax)

    /* Calculate next process. */
    ...
    movl ..., current

    /* Load registers of next process. */
    movl current, %eax
    movl (%eax), %esp
    popl %esi; ...; popl %ebx; popl %eax
    iret
```

# Multi-Tasking

*Programm im User-Modus darf Interrupts nicht verhindern können:*

*=> Nutzung der Befehle*

- cli
- popfw, popfl

*im User-Modus nicht erlaubt (= > Exception) bzw. ohne Auswirkung auf Interrupts.*

*Prozess im User-Modus darf nicht direkt auf I/O-Geräte zugreifen.*

*=> Nutzung der Befehle*

- inb, inw, inl
- outb, outw, outl

*im User-Modus nicht erlaubt (= > Exception).*

# Segmentierung

*Prozess im User-Modus darf nicht auf fremde Speicherbereiche zugreifen.*

*=> Nutzung der Befehle*

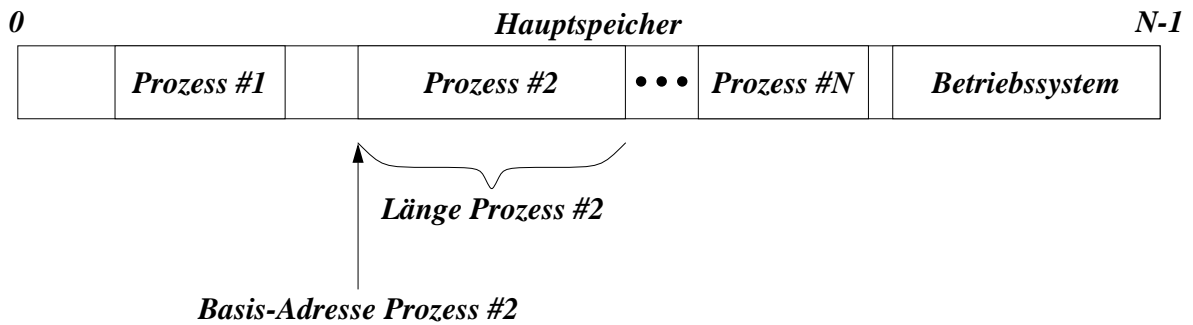
- call, ret
- movb, movw, movl
- addb, addw, addl
- subb, subw, subl
- ...

*im User-Modus nicht erlaubt?!?*

*Diese Lösung ist nicht sinnvoll, da nahezu alle wichtigen Befehle im User-Modus damit unzulässig wären.*

*=> Zugriff auf bestimmte Speicherbereiche beschränken.*

# Segmentierung

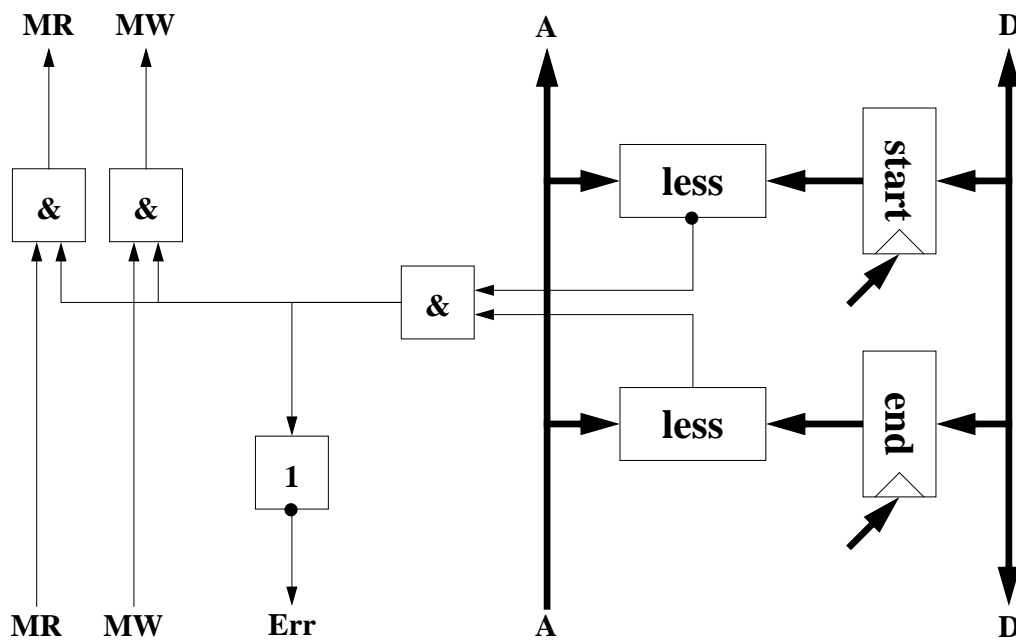


**Für Zugriffsbeschränkung notwendig:**

- Basis- oder Start-Adresse
- Länge (oder End-Adresse)

**des Prozess-Segmentes.**

# Segmentierung



# Segmentierung

## *Prozess im User-Modus*

- kann durch das Laden der Segment-Register „start“ bzw. „end“ mit entsprechenden Werten im User-Modus auf „seinen“ Speicher beschränkt werden
- kann Segment-Register nicht ändern

## *Prozess im privilegierten Modus (im Betriebssystem)*

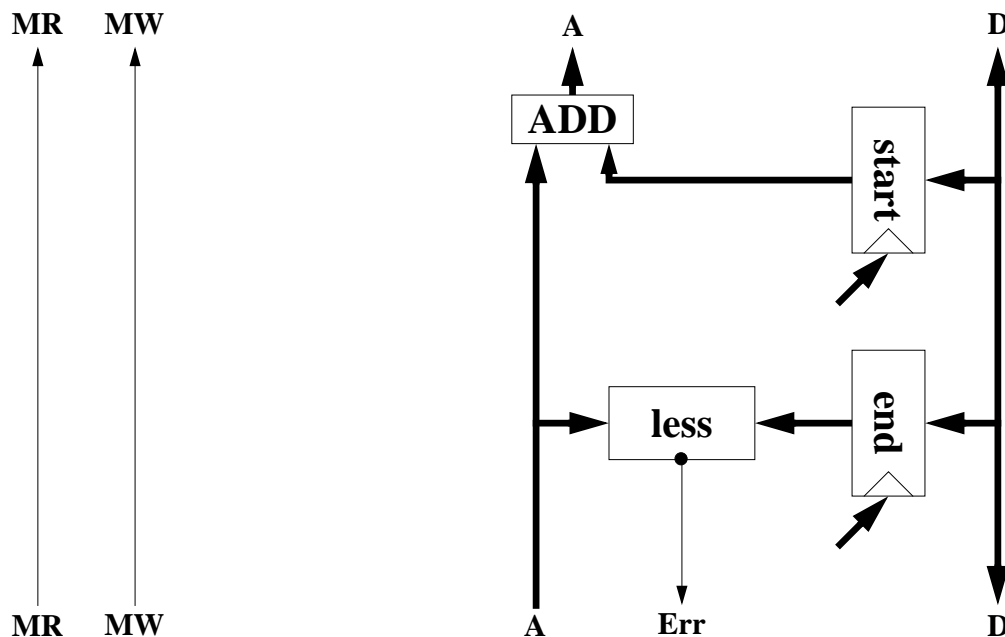
- kann alle Speicherzellen beschreiben / lesen.
- kann Segment-Register neu laden (z.B. bei Prozess-Wechsel)

# Segmentierung

## *Die Segmentierung kann verbessert werden:*

- Start-Adresse aus Sicht des Prozesses im User-Modus ist immer 0.  
=> Unabhängigkeit der Prozesse untereinander verbessert.
- Betriebssystem bestimmt die Lage im Speicher.

# Segmentierung



# Multi-Tasking

*Problem während der Umschaltphase User-Mode -> Priviligierter Modus:*

*Betriebssystem-Segment ist erst nach dem Umladen der Segment-Register zugreifbar.*

*Abspeichern der alten Register-Werte vor dem Umladen nicht möglich.*

*Daher:*

*Interrupt-, Exception- und System-Call-Bearbeitung heißt also:*

- Abspeichern des Instruction-Pointers, Stack-Pointers, Status-Registers, der Segment-Register und des CPL (Current Privilege Level) in temporären Registern.
- Laden der neuen Werte für Instruction-Pointer, Stack-Pointer, Status-Register, der Segment-Register und des CPL.
- Abspeichern der temporären Register auf dem neuen Stack.

# Segmentierung

*Die Segmentierung kann weiter verbessert werden:*

- Mehr Segmente sinnvoll (z.B. Code-Segment, Daten-Segment, Stack-Segment).

Damit werden die Segmente im Speicher kleiner.

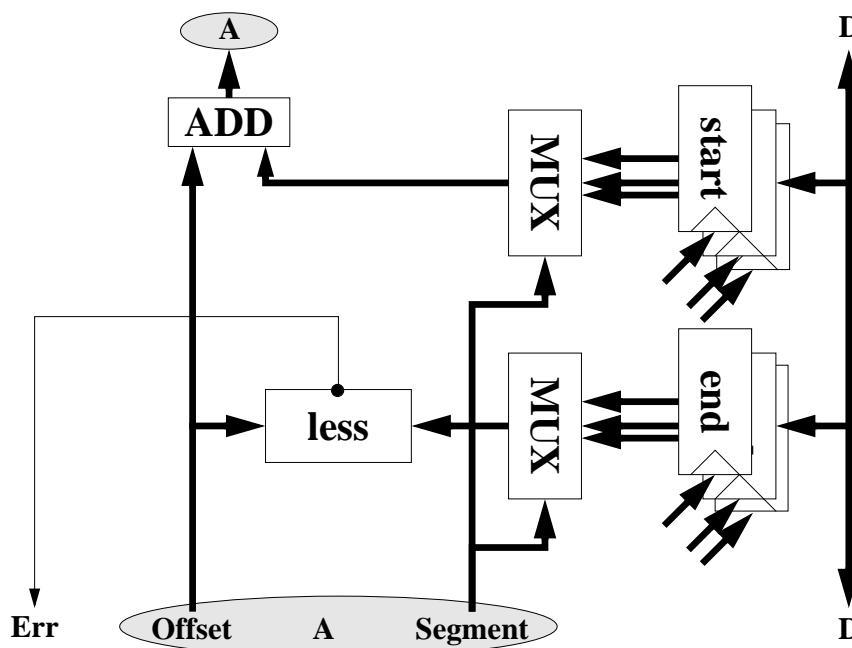
Kleinere Segmente leichter im Speicher einfügbar.

- Attribute für Segmente sinnvoll (z.B. read-only, execute-only, read-write).

*Beispiel Intel-x86: 6 Segmente:*

- %cs: Code-Segment (read-exec)
- %ss: Stack-Segment (read-write)
- %ds, %es, %fs, %gs: Daten-Segmente (read-write)

# Segmentierung



# MMU (Memory-Management-Unit)

Zur leichteren Verteilung der Segmente im Speicher wären kleinere, jeweils gleich große Segmente vorteilhaft (Seiten / Pages).

Programme variieren jeweils im KByte-Bereich => Pages sollten jeweils ca. 1 KByte groß sein.

**Beispiele:**

- Intel: Page-Größe 4 KByte
- Sparc: Page-Größe 8 bzw. 16 KByte

**Problem:**

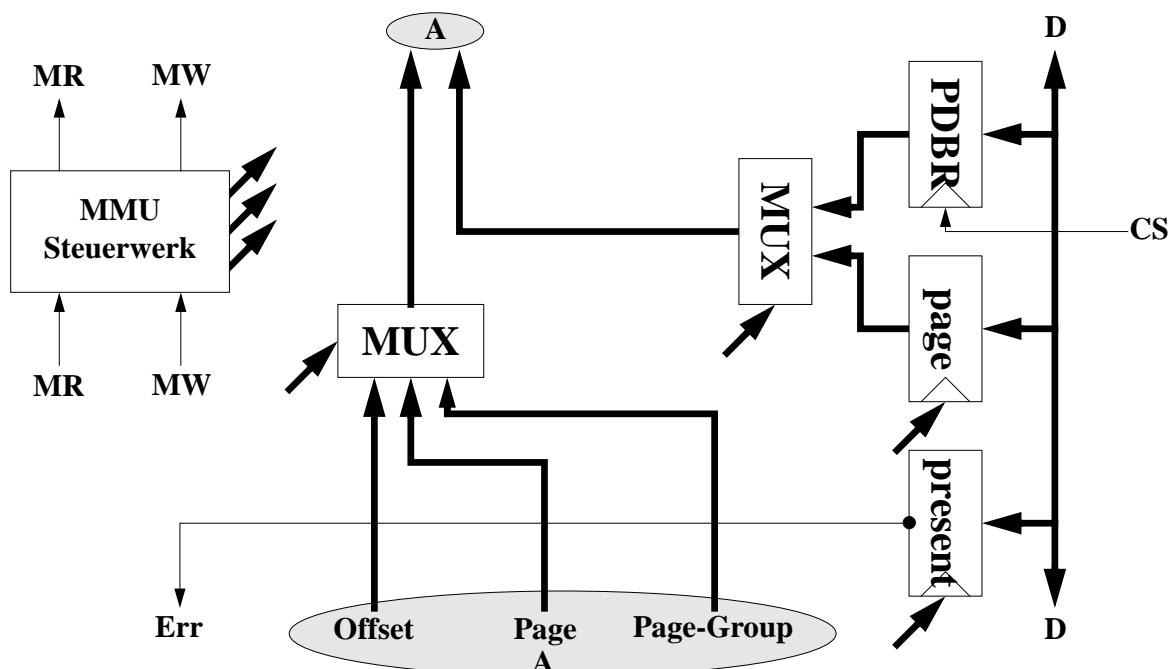
4 GByte Hauptspeicher in 4 KByte Pages aufzuteilen, bedeutet **1 Million** Pages!  
So viele Segment- / Page-Register nicht möglich.

**Lösung:**

Auslagerung der Page-Register in den Hauptspeicher.

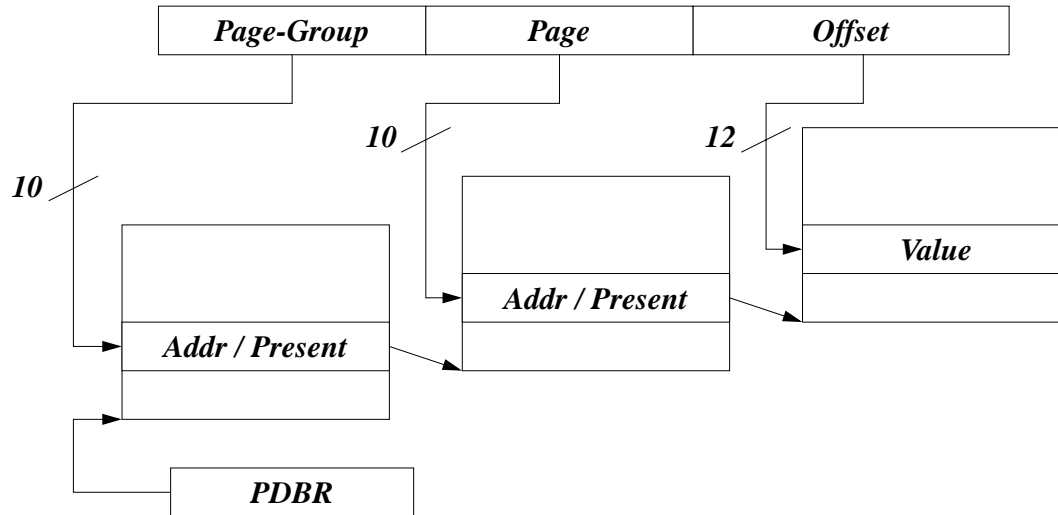
Start-Adresse der Page-Tabellen in einem Register der MMU  
(Page Directory Basis Register; PDBR).

## MMU



# MMU

Umsetzung virtuelle Adresse => physikalische Adresse im Beispiel dreistufig (z.B. I386).



# MMU

**Beispiel:**

**table-Register: 0x00001---**

0x00001000:	0x12000	Start 0x12000000	present
<u>0x00001004:</u>	0x12001--1	Start 0x12001000	present
0x00001008:	0x00000--0		not-present
0x0000100c:	0x13000--1	Start 0x13000000	present
0x00001010:	0x00000--0		not-present
...			
0x00001ffc:	0x00000--0		not-present
0x12001000:	0x00500--1	Start 0x00500000	present
<u>0x12001004:</u>	0x00501--1	Start 0x00501000	present
...			

**Zugriff auf virtuelle Adresse 00000000.01 000000.0001 0011.00110000**

**(page-group 000000001, page 000000001, offset=001100110000)**

**=> Zugriff auf physikalische Adresse 00000000.01010000.0001 0011.00110000**

# MMU

*Umsetzung aufwändig (3 Speicherzugriffe statt einem)!*

**Beobachtung:**

**Speicherlokalität: Programme greifen häufig auf benachbarte Speicherzellen zu:**

- sequentieller Programmablauf
- sequentieller Array-Zugriff
- ...

**=> die N zuletzt gelesenen Tabellenwerte merken (Translation-Lookaside-Buffer; TLB).**

# CPU / MMU

