Script **generated by TTT**

Title: Grundlagen_Betriebssysteme (14.12.2012)

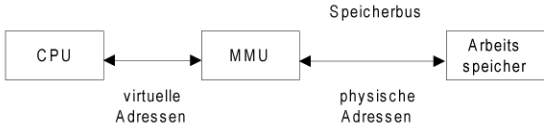
Date: Fri Dec 14 08:30:38 CET 2012

Duration: 91:15 min

Pages: 29

Memory Management Unit

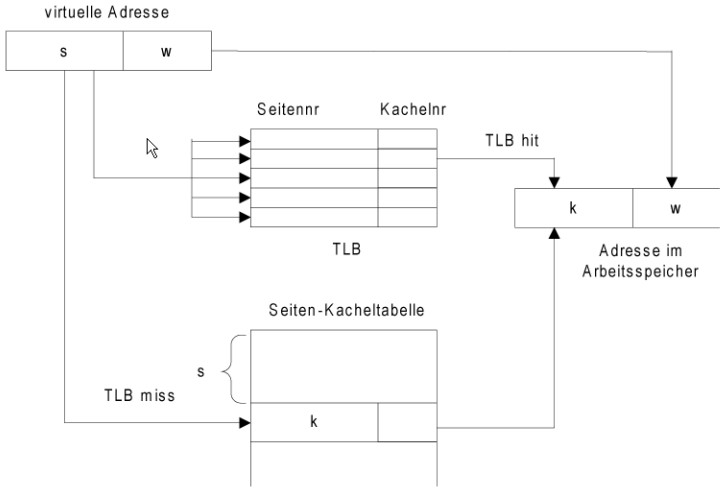
Die Adressrechnung wird von der Hardware, der ***MMU*** (Memory Management Unit), durchgeführt.



Meist ist die MMU auf dem CPU-Chip integriert.

Translation Lookaside Buffer

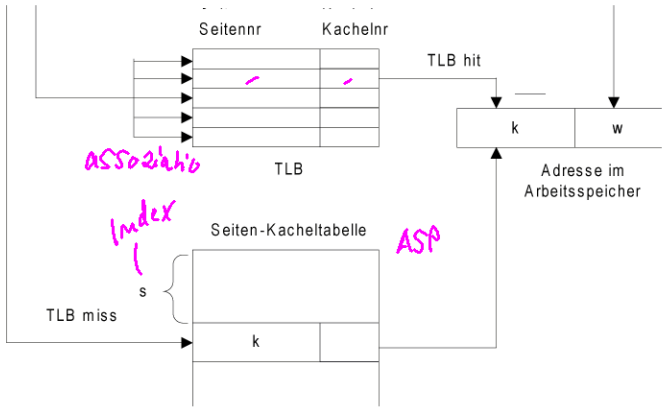
Zur Adressumsetzung enthält die MMU die Tabelle ***Translation Lookaside Buffer*** (TLB), die häufig auftretenden Seitennummern die Kachelnummer zuordnet. TLB umfasst wenige Einträge (z.B. 8, 16, oder 32) und kann als Cache interpretiert werden.



Zugriffszeit

Sei

Translation Lookaside Buffer



Zugriffszeit

Sei

TLB Suche: 20 ns

Speicherzugriff: 100 ns

Bei einem TLB hit ergibt sich Zugriffszeit 120 ns und bei einem TLB miss sind dies 220 ns.

Bei einer TLB-Treffer Wahrscheinlichkeit von 80% ergibt sich als durchschnittliche effektive Speicherzugriffszeit

0,80 * 120 + 0,20 * 220 = 140 ns

Die Adressabbildung erfolgt mittels der ***Seiten-Kacheltabelle*** (SKT). Wir benötigen Informationen über die zu verwaltenden Seiten. Dazu wird jede Seite eines Prozesses durch einen ***Seitendeskriptor*** beschrieben. Die Seiten-Kacheltabelle ist i.a. prozessspezifisch.

Informationen in einem Seitendeskriptor:

Zugriffsrechte (A)

Angabe, ob der Prozess

- ausführend ($x \in A$),
- lesend ($r \in A$),
- schreibend ($s \in A$).

auf die Seite zugreifen darf. Diese Information wird vom Betriebssystem eingetragen und in der CPU vor dem Zugriff ausgewertet.

Seite existent (e)

Seite geladen (v)

Die Abbildung ist gültig (engl. valid). Die Seite ist also in den Arbeitsspeicher geladen, d.h. ihr ist eine Kachel zugeordnet. Es kann auf sie zugegriffen werden.

Zugriffsbit (r)

Bei Zugriff auf Seite: Setzen des Zugriffsbits; relevant für Seitenersetzungsalgorithmen.

Veränderungsbit (m)

Die Seite wurde verändert (modifiziert).

Seiten-Kacheltabelle

Die Adressabbildung erfolgt mittels der ***Seiten-Kacheltabelle*** (SKT). Wir benötigen Informationen über die zu verwaltenden Seiten. Dazu wird jede Seite eines Prozesses durch einen ***Seitendeskriptor*** beschrieben. Die Seiten-Kacheltabelle ist i.a. prozessspezifisch.

Struktur eines Seitendeskriptor

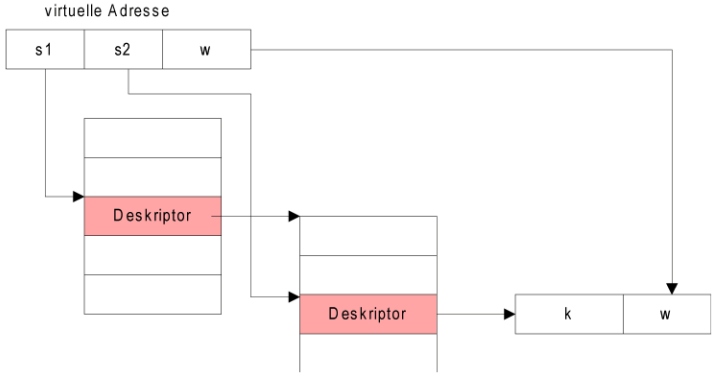
Probleme

Varianten der SKT

Größe der Seiten-Kacheltabelle

Die Seiten-Kacheltabelle kann sehr groß werden, z.B. bei Seitengröße 4K (früher üblich!) ergeben sich für 32-bit Adressraum insgesamt 1 Million Seiten pro Prozess.

Lösung? ***Mehrstufige Tabellen*** : Seiten-Kacheltabelle ebenfalls in Seiten unterteilen, die ihrerseits in Tabellen gehalten werden.



Linux verwendet ein 3-stufiges Paging Verfahren

jede virtuelle Adresse ist in 4 Teile aufgeteilt.

Es gibt einige Probleme mit der Seitenadressierung, die sich auf die Speichereffizienz und die Performanz beziehen.

Größe der Seiten-Kacheltabelle

Performanz

Schnelle Umrechnung der virtuellen auf realen Adressen erforderlich, da Berechnung ***bei jedem Zugriff*** notwendig ist! Häufig verwendete Adressen werden in einem Schnellzugriffspeicher (cache) der MMU gehalten, dem ***Translation Lookaside Buffer*** (TLB).

Varianten für den Aufbau einer Seiten-Kacheltabelle.

Prozess-spezifisch, Index-basiert (PI)

Zugriff auf Deskriptor für Seite s_i über Index i.

Prozess-spezifisch, assoziativ (PA)

Hier wird die Seitennummer mit in den Seitendeskriptor aufgenommen.

Global, assoziativ (GA)

Es gibt nur eine SKT im System. Da die Seitennummer allein nicht eindeutig ist, muss das Prozesskennzeichen (PID), d.h. der eindeutige Prozessname, ebenfalls in den Deskriptor aufgenommen werden.

Global, indiziert (GI)

Enthält auch PID und es erfolgt eine Reihung gemäß der Kachelnummer; freie Kacheln werden durch einen speziellen Eintrag gekennzeichnet.

Seiten-Kacheltabelle

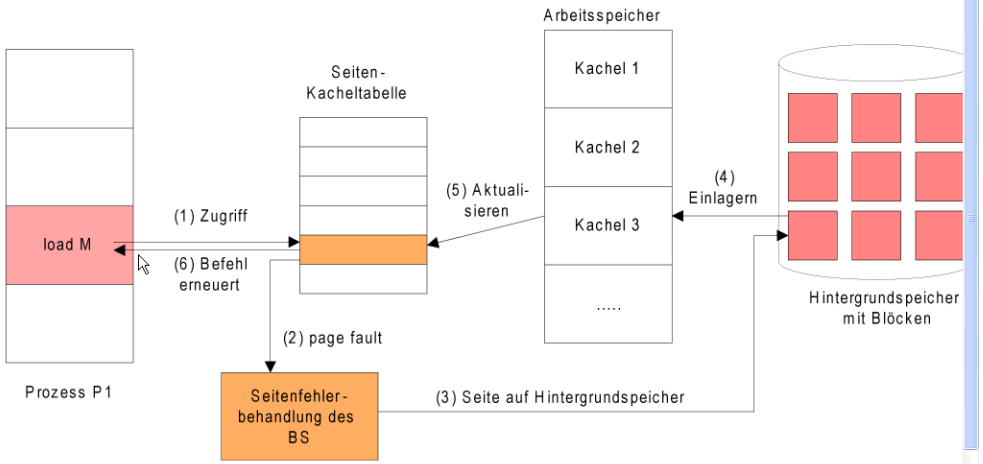
Die Adressabbildung erfolgt mittels der ***Seiten-Kacheltabelle*** (SKT). Wir benötigen Informationen über die zu verwaltenden Seiten. Dazu wird jede Seite eines Prozesses durch einen ***Seitendeskriptor*** beschrieben. Die Seiten-Kacheltabelle ist i.a. prozessspezifisch.

Struktur eines Seitendeskriptor

Probleme

Varianten der SKT

Der Zugriff auf eine Seite, die nicht im Arbeitsspeicher ist, führt zu einem Seitenfehler.



Ablauf der Seitenfehlerbehandlung

1. Beim Zugriff auf eine virtuelle Adresse (z.B. LOAD-Befehl) tritt ein Seitenfehler auf.
2. Die Adressrechnungshardware löst einen Alarm aus, so dass die Unterbrechungsbehandlung des BS aktiviert wird.

Seitenadressierung

Die virtuelle Adressierung wurde Ende der 50er Jahre eingeführt. Ziel ist

- Virtualisierung des Speichers,
- Verstecken von realen Beschränkungen, wie Speichergröße,
- Speicher als sehr großes Feld gleichartiger Speicherzellen zu betrachten.

Die Seitenadressierung ("paging") ist die Grundform der virtuellen Adressierung.

Aufgabe der Arbeitsspeicherverwaltung: Laden der für die Ausführung der Prozesse benötigten Seiten in die Kacheln. Es ergeben sich drei strategische Aufgaben:

Platzierungsstrategie

Frage: in welche Kachel ist eine Seite zu Laden?

Lösung

keine strategische Entscheidung erforderlich, da alle Kacheln gleichwertig sind und damit keine Auswahl getroffen werden muss. Vorteil der uniformen Realisierungskonzepte (Seite, Kachel).

Frage: welche Seite ist zu Laden?

Lösungsansätze

Einzelseitenanforderungsstrategie (***on demand***): eine Seite wird genau dann geladen, wenn auf sie zugegriffen wird und sie sich noch nicht im Arbeitsspeicher befindet.

Seiten-***Prefetching***: Seiten werden im Voraus geladen, um sie sofort bei Bedarf verfügbar zu haben.

Windows XP kombiniert "Demand Paging" mit "Prefetching": bei einem Seitenfehler werden die gewünschte Seite sowie auch einige nachfolgende Seiten (falls möglich) in den Arbeitsspeicher eingelagert.

Aufgabe der Arbeitsspeicherverwaltung: Laden der für die Ausführung der Prozesse benötigten Seiten in die Kacheln. Es ergeben sich drei strategische Aufgaben:

Ladestrategie

Platzierungsstrategie

Frage: in welche Kachel ist eine Seite zu Laden?

Lösung

keine strategische Entscheidung erforderlich, da alle Kacheln gleichwertig sind und damit keine Auswahl getroffen werden muss. Vorteil der uniformen Realisierungskonzepte (Seite, Kachel).

Seitenverdrängungsstrategie

Weitere offene Fragen

Generated by Targeteam

Frage: welche Seite ist aus dem Arbeitsspeicher zu entfernen, wenn für eine zu ladende Seite keine freie Kachel mehr zur Verfügung steht?

FIFO Strategie

Second-Chance

Variante von FIFO, die vermeidet, dass ältere Seiten, obwohl sie häufig benutzt werden, ausgelagert werden. Neben dem Status "älteste Seite" wird auch Zugriffsbit r betrachtet:

r = 1: Seite wird an das Ende der FIFO Liste gestellt; r wird auf 0 gesetzt.

r = 0: Seite wird ausgelagert; bei Modifikation (m-Bit gesetzt) wird Seite zurückgeschrieben.

Clock-Algorithmus

Weitere Strategien

FIFO (first-in first-out): Verdrängen der ältesten Seite, einfach zu implementieren. Das Verfahren berücksichtigt die Lokalitätsmenge nicht.

Belady's Anomalie

Zuordnung von zusätzlichen Kacheln an einen Prozess reduziert nicht notwendigerweise die Anzahl der Seitenfehler. Das Gegenteil ist möglich. Beispiel:

Aufrufsequenz von benötigten Seitennummern: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

jeweils 3 Seiten im Arbeitsspeicher:

(1, 2, 3), (2, 3, 4), (3, 4, 1), (4, 1, 2), (1, 2, 5), (2, 5, 3), (5, 3, 4) ⇒ 9 Seitenfehler

jeweils 4 Seiten im Arbeitsspeicher:

(1, 2, 3, 4), (2, 3, 4, 5), (3, 4, 5, 1), (4, 5, 1, 2), (5, 1, 2, 3), (1, 2, 3, 4), (2, 3, 4, 5) ⇒ 10 Seitenfehler

Seitenverdrängungsstrategie

Frage: welche Seite ist aus dem Arbeitsspeicher zu entfernen, wenn für eine zu ladende Seite keine freie Kachel mehr zur Verfügung steht?

FIFO Strategie

Second-Chance

Variante von FIFO, die vermeidet, dass ältere Seiten, obwohl sie häufig benutzt werden, ausgelagert werden. Neben dem Status "älteste Seite" wird auch Zugriffsbit r betrachtet:

r = 1: Seite wird an das Ende der FIFO Liste gestellt; r wird auf 0 gesetzt.

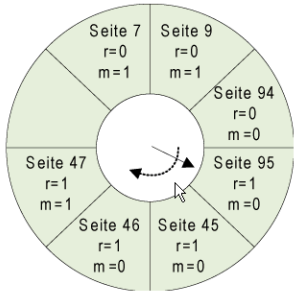
r = 0: Seite wird ausgelagert; bei Modifikation (m-Bit gesetzt) wird Seite zurückgeschrieben.

Clock-Algorithmus

Weitere Strategien

Eine Implementierungsart von Second-Chance ist der ***Clock-Algorithmus*** . Die Seiten werden in einem Ring angeordnet.

Oft in Kombination mit dem m-Bit (Modifikation).



Verfahren

1. beginnend von der aktuellen Pointerposition suche die erste Seite mit (r=0, m=0); diese Seite wird ausgelagert.
2. Falls 1. Schritt fehlschlägt, suche nach der ersten Seite mit (r=0; m=1); diese Seite wird ausgelagert. Von untersuchten Seite wird das r-Bit zurückgesetzt.
3. Falls 2. Schritt fehlschlägt, starte von der ursprünglichen Pointerposition; wiederhole den 1., und falls notwendig, den 2. Schritt.

Seitenverdrängungsstrategie

Frage: welche Seite ist aus dem Arbeitsspeicher zu entfernen, wenn für eine zu ladende Seite keine freie Kachel mehr zur Verfügung steht?

FIFO Strategie

Second-Chance

Variante von FIFO, die vermeidet, dass ältere Seiten, obwohl sie häufig benutzt werden, ausgelagert werden. Neben dem Status "älteste Seite" wird auch Zugriffsbit r betrachtet:

r = 1: Seite wird an das Ende der FIFO Liste gestellt; r wird auf 0 gesetzt.

r = 0: Seite wird ausgelagert; bei Modifikation (m-Bit gesetzt) wird Seite zurückgeschrieben.

Clock-Algorithmus

Weitere Strategien

Working-Set ist die Menge der Seiten, die ein Prozess zu einem bestimmten Zeitpunkt benutzt.

w(k,t) = Menge der Seiten, die zum Zeitpunkt t in den letzten k Speicherzugriffen benutzt wurden.

die Zahl k kann dynamisch angepasst werden.

Verdrängungsstrategie: ersetze eine der Seiten, die nicht zum aktuellen Working-Set gehören.

Verdrängung einer Seite notwendig, obwohl alle Seiten zu w(k,t) gehören:

verkleinere den Parameter k

Reduziere die Anzahl der Prozesse im Arbeitsspeicher ⇒ Auslagern eines Prozesses auf die Festplatte (Swapping).

Working-Set kann nur näherungsweise über Zugriffsbits (r) und Veränderungsbit (m) bestimmt werden.

Weitere Strategien

LIFO (last-in first-out): Verdrängen der jüngsten Seite, einfach zu implementieren.

LRU (Least recently used): Verdrängen der am längsten nicht genutzten Seite; wird am häufigsten realisiert, wobei LRU approximiert wird, da eine exakte Realisierung zu aufwendig ist.

Working-Set Modell

Optimale Strategie : Seite, auf die in Zukunft am längsten nicht zugegriffen wird.

Seitenverwaltungsstrategien

Aufgabe der Arbeitsspeicherverwaltung: Laden der für die Ausführung der Prozesse benötigten Seiten in die Kacheln. Es ergeben sich drei strategische Aufgaben:

Ladestrategie

Platzierungsstrategie

Frage: in welche Kachel ist eine Seite zu Laden?

Lösung

keine strategische Entscheidung erforderlich, da alle Kacheln gleichwertig sind und damit keine Auswahl getroffen werden muss. Vorteil der uniformen Realisierungskonzepte (Seite, Kachel).

Seitenverdrängungsstrategie

Weitere offene Fragen

Wahl einer vernünftigen Seitengröße

Die Wahl der Seitengröße steht unter widersprüchlichen Zielsetzungen; daher ist ein Kompromiss erforderlich.

1. Je kleiner die Seite, desto rascher die Transfers zwischen ASP und Platte.
2. Je kleiner die Seite, desto geringer der Verschnitt (interne Fragmentierung) durch nicht voll ausgenützte Seiten.
3. Je größer die Seite, desto geringer der Overhead für Transport zwischen Arbeitsspeicher und Platte pro Byte (Arm positionieren, Warten bis Spur unter Lese-Schreib-Kopf).
4. Je größer die Seite, desto mehr Information kann zwischen Arbeitsspeicher und Platte je Zeiteinheit transportiert werden.
5. Je größer die Seite, desto seltener Transfers erforderlich.

Beispiele für Seitengrößen: Intel 80386 4K, Pentium II 4K oder 4MB, UltraSPARC 8K, 64K, 512K oder 4MB.

Seitenflattern

Seitenflattern (engl. thrashing) tritt auf, wenn im System zuviele Prozesse sind, die nebenläufig voranschreiten wollen und Kacheln beanspruchen. Gerade verdrängte Seiten müssen zu schnell wieder eingelagert werden, das System ist im schlimmsten Fall nur noch mit Ein- und Auslagern beschäftigt.

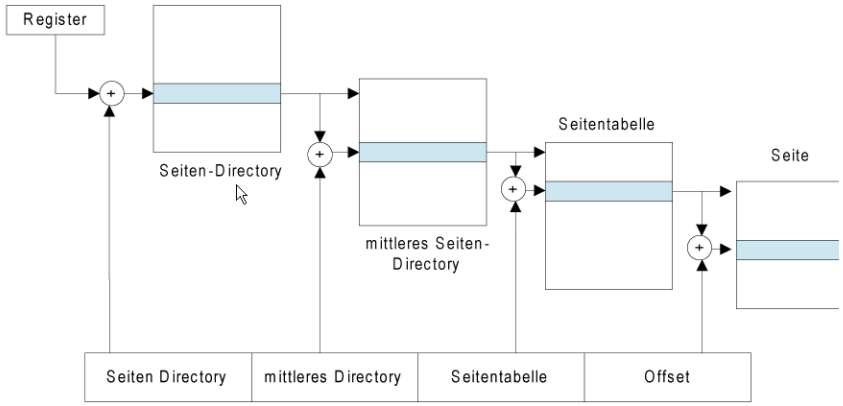
Linux - Virtuelle Adressierung

Die virtuelle Adressierung in Linux basiert auf mehrstufigen Seitentabellen

Seiten-Directory ("page directory").

mittleres Seiten-Directory ("page middle directory").

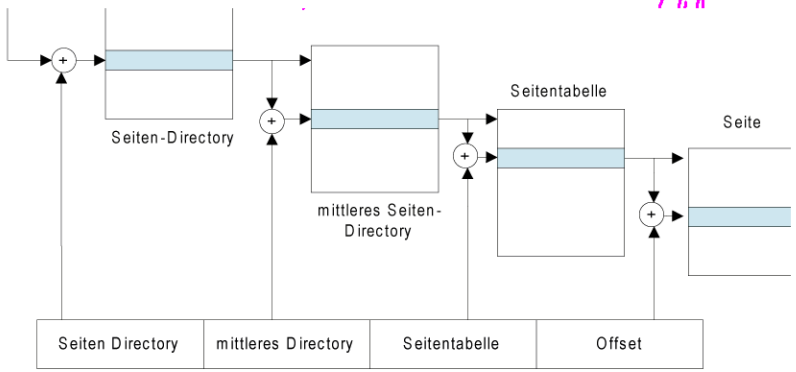
Seitentabelle ("page table").



Seitenallokation

aufeinanderfolgende Seiten werden auf zusammenhängende Kacheln abgebildet.

Linux - Virtuelle Adressierung



Seitenallokation

aufeinanderfolgende Seiten werden auf zusammenhängende Kacheln abgebildet.

Behandlung von Gruppen mit 1, 2, 4, 8, 16, oder 32 Seiten.

Seitenverdrängung

Anwendung eines modifizierten Clock-Algorithmus. Zugriffsbit r wird durch einen 8-bit Zähler ersetzt.

bei jedem Zugriff wird der Zähler inkrementiert.

Linux dekrementiert periodisch die Zähler aller Seiten im Arbeitsspeicher.

Unterteilung des Programmadressraums in logische Einheiten unterschiedlicher Länge, sogenannte ***Segmente*** . Ein Segment umfasst inhaltlich bzw. organisatorisch zusammengehörige Speicherbereiche, z.B. Daten, Code und Laufzeitkeller-Segment.

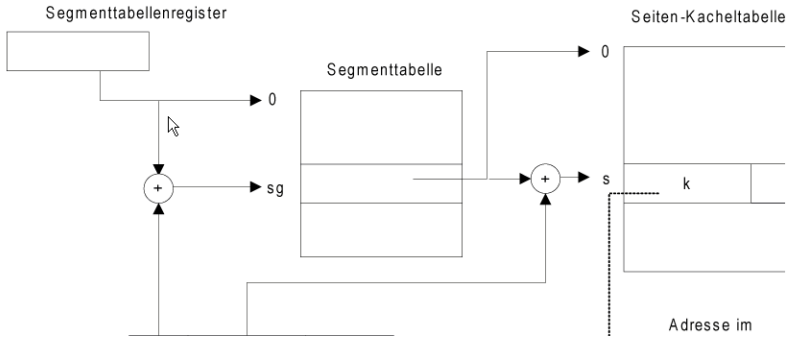
Jedes Segment besitzt eine maximale Größe.

Jedes Segment wird durch einen Segment-Deskriptor beschrieben. Die Segmentdeskriptoren werden in einer Segmenttabelle verwaltet.

Jedes Segment besteht aus Seiten, die jeweils beginnend mit Null fortlaufend numeriert sind.

Ein Zugriff auf ein nicht existentes Segment führt zum Speicherschutzalarm.

Um in dieser Situation möglichst kompakte Speicherabbildungstabellen zu erhalten, wird die Seiten-Kacheltabelle aufgeteilt. ⇒ je Segment eine eigene Seiten-Kacheltabelle gehalten. Die (Maschinen-) Adressen der Seiten-Kacheltabellen werden in einer Segmenttabelle gehalten.



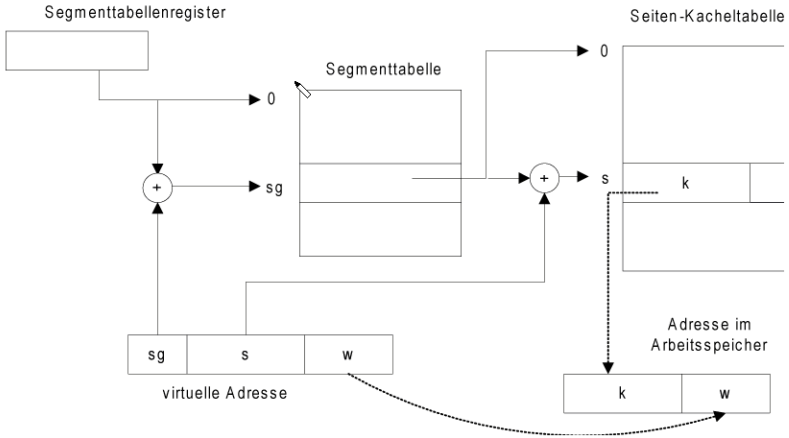
Segment-Seitenadressierung

Seitenteilung systembedingt

Jedes Segment besteht aus Seiten, die jeweils beginnend mit Null fortlaufend numeriert sind.

Ein Zugriff auf ein nicht existentes Segment führt zum Speicherschutzalarm.

Um in dieser Situation möglichst kompakte Speicherabbildungstabellen zu erhalten, wird die Seiten-Kacheltabelle aufgeteilt. ⇒ je Segment eine eigene Seiten-Kacheltabelle gehalten. Die (Maschinen-) Adressen der Seiten-Kacheltabellen werden in einer Segmenttabelle gehalten.



Fragestellungen

Dieser Abschnitt beschäftigt sich mit den Adressräumen für Programme und deren Abbildung auf den physischen Arbeitsspeicher einer Rechenanlage:

- Programmadressraum vs. Maschinenadressraum.
- Direkte Adressierung, Basisadressierung.
- Virtualisierung des Speichers; virtuelle Adressierung, insbesondere Seitenadressierung.