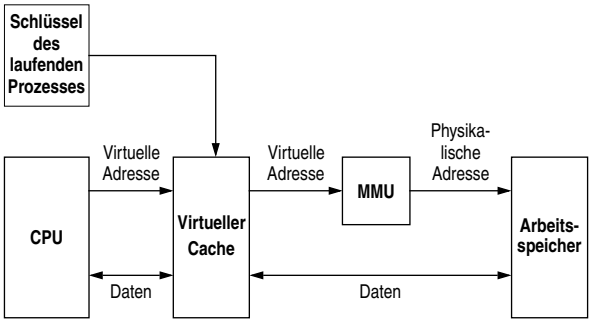
4.4 Virtuelle Caches mit Prozeß-Schlüsseln

4.4.1 Arbeitsweise



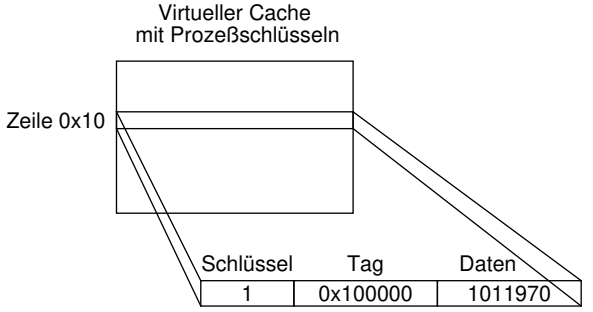
© 1997, Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

4.4.2 Verwaltung

- ❏ Kontextumschaltung
  - bei write-through kein Leeren bei Kontextumschaltung erforderlich
  - bei write-back Schwierigkeiten, da nach Kontextwechsel die alte, zuständige SKT nicht mehr bekannt ist
  - bei zu wenig Schlüsselwerten muß die Prozessorvergabe auf die Schlüsselverteilung Rücksicht nehmen, wenn nicht jeder Kontextwechsel zur Leerung des Cache führen soll

© 1997, Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

- ❏ fork
  - Sohn braucht neuen Schlüssel
  - Daten des Vaters müssen im Arbeitsspeicher sein!
  - kein copy-on-write
    - write-through: Kern-Adreßraum muß nicht bereinigt werden
    - write-back: Cache muß explizit geleert werden
  - copy-on-write
    - write-back: Arbeitsspeicher muß bei jedem fork validiert werden, damit Zeilenersetzungen kein copy-on-write-Signal auslösen
    - Beispiel
      Schlüssel des Vaters: 1
      Schlüssel des Sohnes: 2
      Cache direkt abgebildet, write-through
      Vater liest:



© 1997, Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

Wenn Sohn liest, wird trotz copy-on-write erneut vom Speicher gelesen

Abhilfe: mehrfach abgebildeter Cache
Beispiel 2-fach

Nur Vater hat aus virtueller Adresse 0x100000 gelesen:

Menge 0x10

Schlüssel	Tag	Daten
1	0x100000	1011970
-	-	-

Nur Sohn hat aus virtueller Adresse 0x100000 gelesen:

Menge 0x10

Schlüssel	Tag	Daten
2	0x100000	1011970
-	-	-

Vater und Sohn haben aus virtueller Adresse 0x100000 gelesen:

Menge 0x10

Schlüssel	Tag	Daten
1	0x100000	1011970
2	0x100000	1011970

© 1997, Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

❑ Exec

- Prozeßschlüssel beibehalten
- im Cache alle Einträge mit diesem Schlüssel invalidieren

❑ exit

da normalerweise kein Flush bei Prozeßumschaltung, Einträge diese Schlüssels invalidieren

könnte auch bis zur Wiedervergabe des Schlüssels aufgeschoben werden

❑ brk, sbrk

wie bei virtuellem Cache

❑ shared memory und momory mapped files

Information könnte mehrfach unter verschiedenen Schlüsseln vorhanden sein

- unter gleicher virtueller Adresse problematisch, falls nicht write-allocate

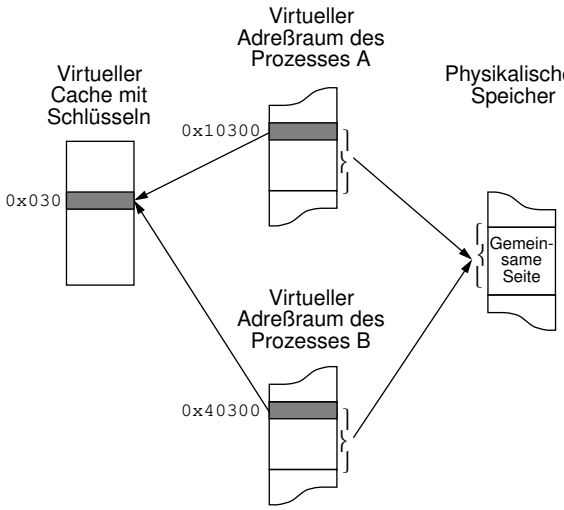
Schlüssel	Mod	Tag	Daten
1	M	0x100000	1011970

Nach Zugriffsfehler:

Schlüssel	Mod	Tag	Daten
2	-	0x100000	1011970

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

- unter verschiedenen virtuellen Adressen



Möglichkeiten zur Verhinderung von Mehrdeutigkeiten

- Entfernung aus dem Cache bei Adreßraumumschaltung
- Puffern (“cachen”) gemeinsamer Segmente nicht zulassen
- Bei direkt abgebildetem Cache mit write_allocate, gemeinsame Segmente so anlegen, daß Anfangsadressen auf die gleiche Cachezeile abgebildet werden; aber Zugriff erzeugt Fehlzugriff!

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

❑ E/A

zu behandeln wie rein virtueller Cache

❑ Mehrdeutigkeiten Benutzer - Kern

- zusätzlicher Modus-Indikator
- besonderer Schlüssel für Betriebssystem
  - Vorteil: Kern hat Zugang zu seinen Daten unabhängig von der Identität des aufrufenden Prozesses
  - Nachteil: Fehlzugriffe bei Zugriff auf Benutzerdaten

4.4.3 Benutzung virtueller Caches in MMUs

❑ Caches werden in MMUs benutzt als

- ‘translation lookaside buffer’ (TLB), auch als ‘address translation cache’ (ATC) bezeichnet
- TLB ist virtueller Cache, der Kacheladressen (also Daten der Seiten-Kachel-Tabelle) und Zugriffsrechte speichert
- Fehlzugriffe werden in manchen Systemen nicht durch Hardware sondern durch BS behandelt (z. B. MIPS)
- als TLBs werden häufig virtuelle Caches mit Schlüsseln verwendet; vorteilhaft, wenn (wie üblich) SKTs keine gemeinsamen Abschnitte haben

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig

4.4.4 Schlußfolgerung

❑ virtuelle Caches mit Schlüsseln reduzieren Zusatzaufwand: Cache muß normalerweise bei Prozeßumschaltung nicht invalidiert werden.

❑ Nachteil: gemeinsame Daten können im Cache nicht als solche benutzt werden; verbesserbar, wenn BS Anfangsadressen vergeben kann, da dann bei fork (aber auch nur bei fork!) auf explizite Invalidierung des Cache verzichtet werden kann

❑ zusätzliche Probleme treten auf, wenn nicht genügend Prozeßschlüssel zur Verfügung stehen

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors nicht zulässig