Rechnerstrukturen

5. Speicher

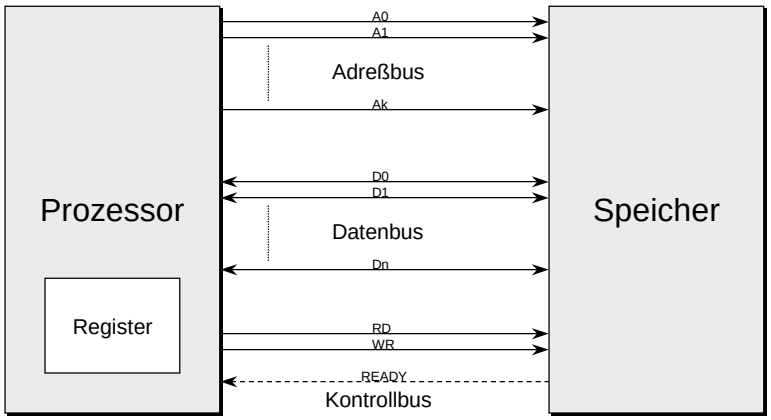
5.1

Inhalt

- Motivation
- Speichertypen
  - RAM / ROM
  - Dynamisches RAM
- Cache-Speicher
  - Voll Assoziativ
  - n-Wege Assoziativ
  - Direct Mapping
- Virtueller Speicher

5.2

Der Hauptspeicher



5.3

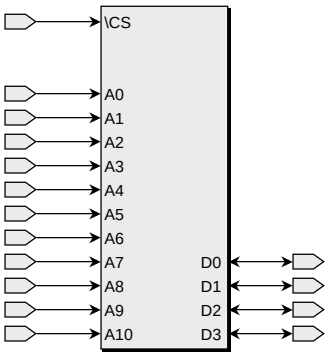
Speichertypen

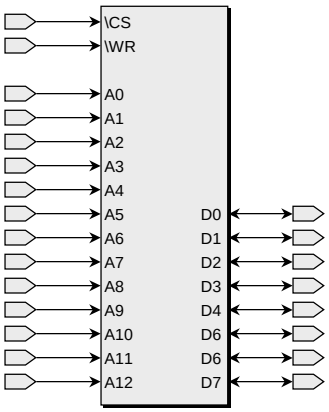
- Lese- und Schreibspeicher (RAM)
  - Flüchtig
  - Statische RAM (Flip-Flop)
  - Dynamische RAM (Kondensatoren)
  - Statisch-dynamische RAM
    - Kondensator im Flip-Flop-Pelz
- Nurlesespeicher (ROM)
  - Nicht-flüchtig
  - PROM = Einmal programmierbar
  - EPROM = Erasable ROM (UV-Licht)
  - EEPROM = Electrically erasable ROM
  - Flash ROM = “Fast ein nicht-flüchtiges RAM”

5.4

Beispiel ROM und SRAM

- 2048 x 4 Bit ROM
- 8128 x 8 Bit statisches RAM

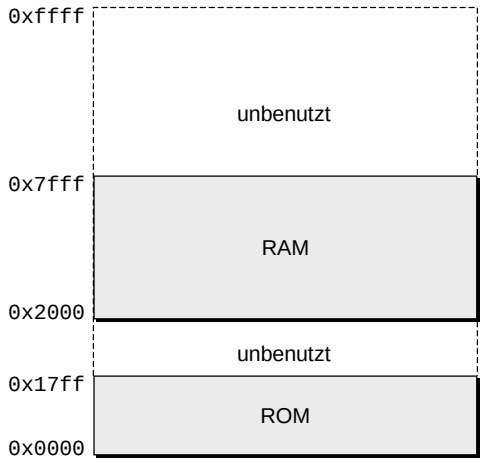




5.5

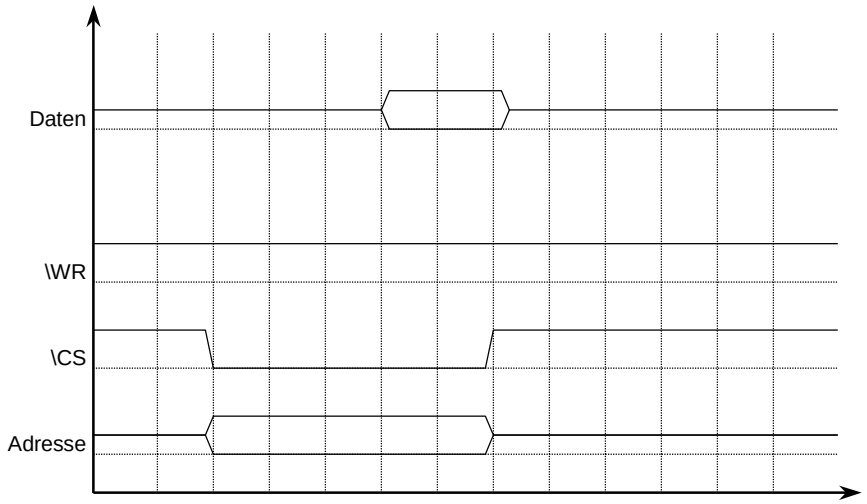
Aufbau eines 64 K x 16 Bit-Speichers?

- Gewünschtes Speicherlayout



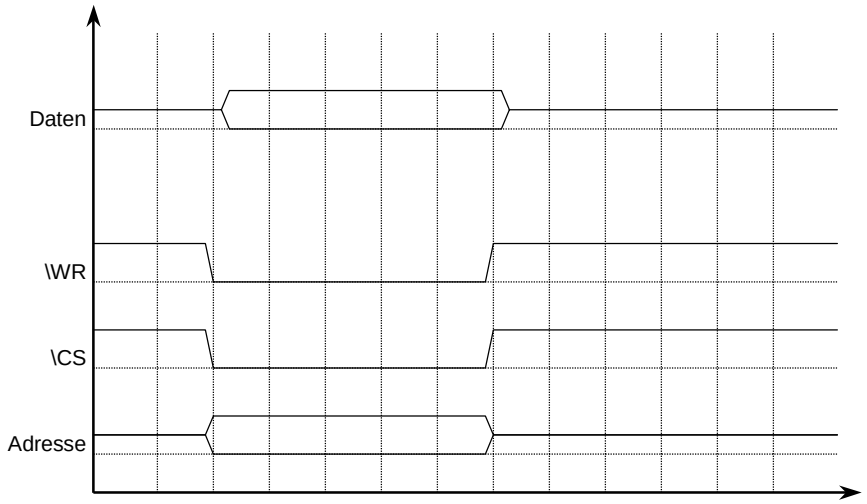
5.6

Lesezyklus



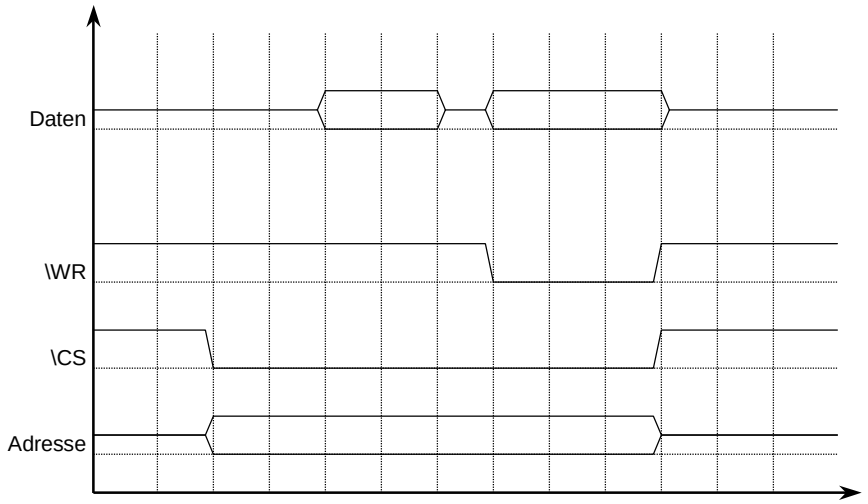
5.7

Schreibzyklus



5.8

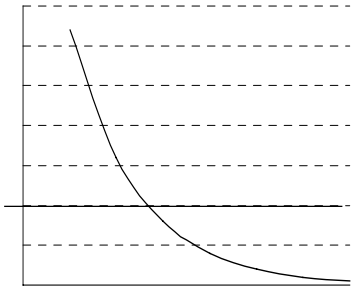
Read-Modify-Write-Zyklus



5.9

Dynamisches RAM (DRAM)

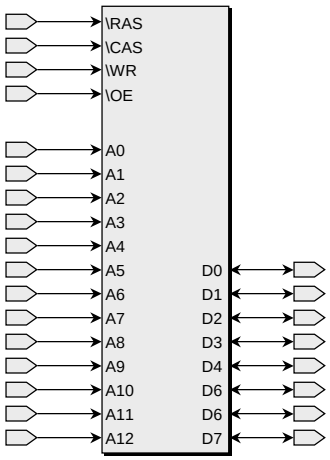
- 1 Kondensator pro Bit
  - Erheblich kleiner als statisches RAM
    - hohe Integrationsdichte
    - geringer Verbrauch
  - Kapazitiv = verhältnismäßig langsam
- Integrationsdichte
  - 64 Mbit-Chips Stand der Technik
  - 256 Mbit-Chips demnächst
  - 1 Gbit-Chips im Labor
- Nachteil
  - Kondensator verliert langsam Ladung
  - Schwellwert für sicheres Erkennen eines 1-Bits
  - Refresh ca. alle 4 bis 64 msec



5.10

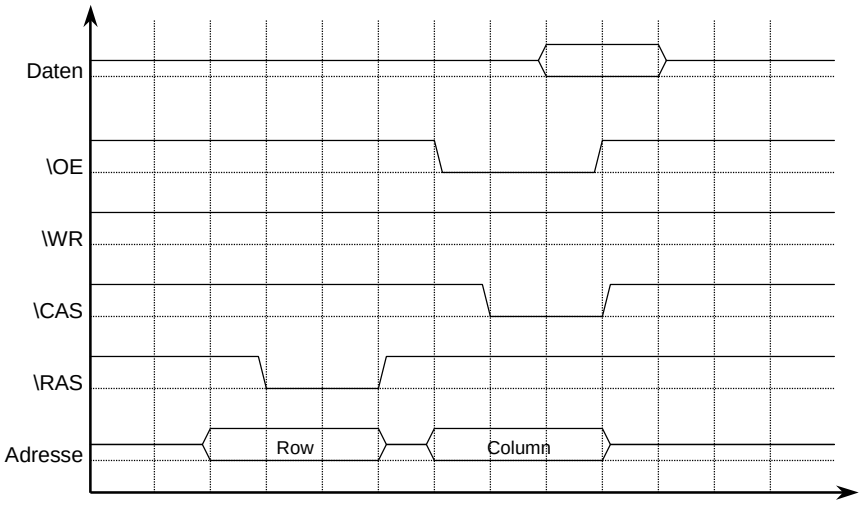
Typische Pin-Belegung

- Viele Adreßleitungen
  - 64 Mbit = 26 Leitungen
  - 256 Mbit = 28 Leitungen
  - 1 Gbit = 30 Leitungen
- Minimierung der Anschlüsse
  - Adresse in zwei Schritten
    - Row
    - Column
  - Adreßleitungen halbiert
  - RAS = Row Address Strobe
  - CAS = Column Address Strobe
- Spezielle Freischaltung der Ausgänge
  - OE = Output Enable



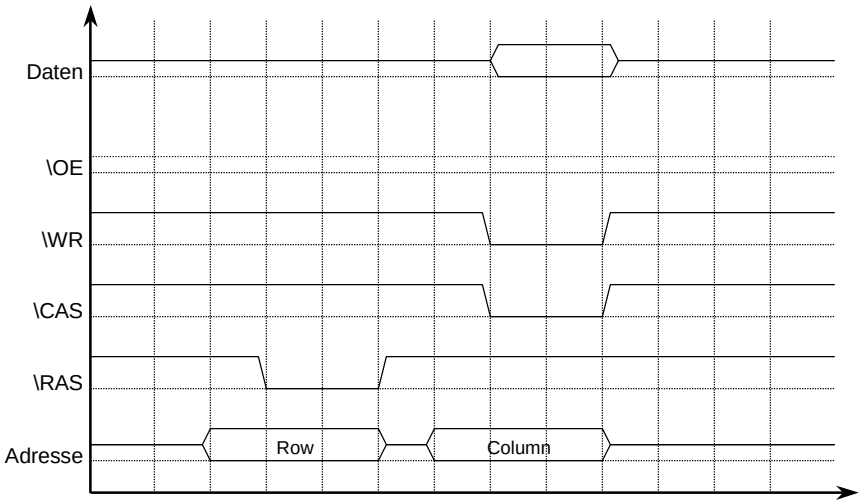
5.11

Lesezyklus



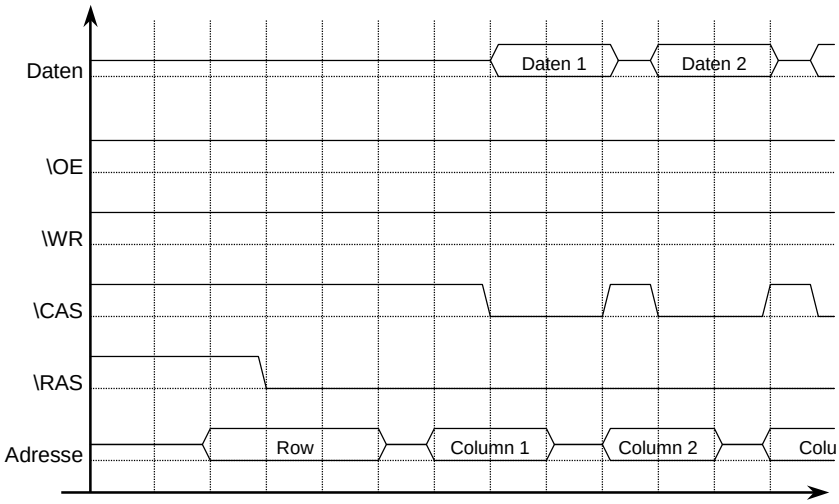
5.12

Schreibzyklus



5.13

Fast-Page-Mode



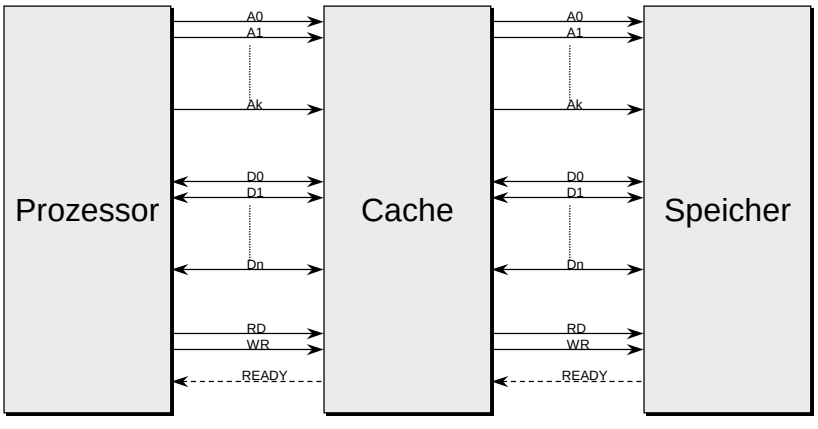
5.14

Refresh

- Bei jedem Zugriff auf eine Reihe mittels \RAS (lesend oder schreibend), wird diese Reihe aufgefrischt
  - Jede Reihe mindestens alle 8 bis 64 msec ansprechen
- Verschiedene Techniken
  - RAS-Only-Refresh
    - Nur RAS-Signal für Refresh
  - CAS-Before-RAS-Refresh
    - Aktiviert internen Row-Zähler
- Modi
  - Burst-Modus: Alle x msec alle Reihen auffrischen
  - Distributed-Modus: Auffrischen einzelner Reihen zeitlich verteilen
  - Hidden-Refresh: Refresh während zugriffsfreier Zeiten
- DRAM-Controller
- SDRAM: DRAM mit internem Controller

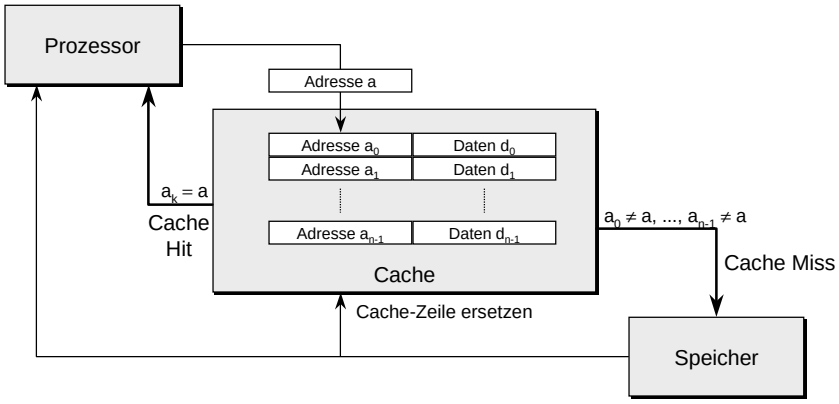
5.15

Der Cache



5.16

Funktionsweise



- Idealerweise gleichzeitiger Vergleich aller a_i mit a
  - Assoziative Suche
- Ersetzungsstrategie
  - Welche Zeile im Cache wird bei Cache Miss ersetzt

5.17

Ziele

- Eigenschaften von Cache und Hauptspeicher:
  - Zugriffsgeschwindigkeit t_C und t_M
  - Kapazität C_C und C_M
  - Preis pro Byte P_C und P_M

$$t_C << t_M, P_C >> P_M, C_C << C_M$$

- Eigenschaften des Cache-Hauptspeicher-Verbund:

$$t_{eff} = p \cdot t_C + (1-p) \cdot t_M, P_{eff} = \frac{P_C \cdot C_C + P_M \cdot C_M}{C_C + C_M}, C_{eff} = C_M$$

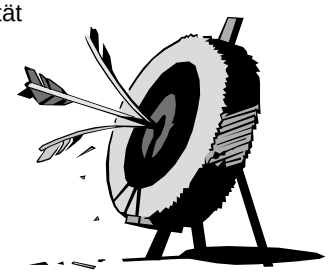
- Ziel

$$t_{eff} \approx t_C \Rightarrow p \approx 1, P_{eff} \approx P_M \Rightarrow C_C << C_M$$

5.18

Trefferrate

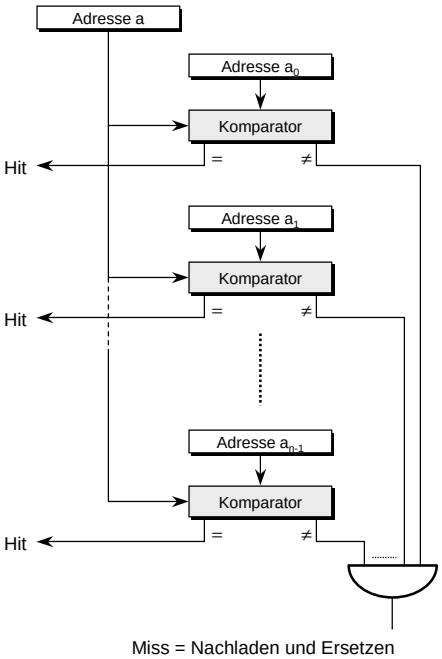
- p = 0.85 - 0.95 wird meist erreicht
- Referenzlokalität
  - Zugriff auf Instruktionen in einer Schleife
  - Sequentieller Zugriff auf Instruktionen und Daten
  - Häufig referenzierte und ev. veränderte Daten
  - ...
- Gilt primär nur für prozedurale und imperative Sprachen
  - Keine so ausgeprägte Referenzlokalität bei funktionalen und logischen Programmiersprachen



5.19

Voll-assoziativer Cache

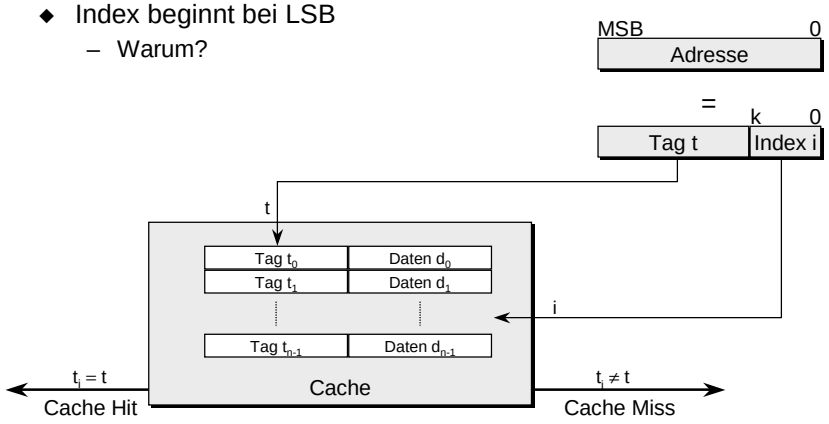
- Hoher Aufwand
  - Komparator für jedes a_k
  - Großer Flächenbedarf
- Keine Duplikate
  - Jede Adresse wird maximal einmal gespeichert
- Ersetzungsstrategie
  - Welche Zeile wird bei Miss ersetzt
  - LRU gängig: Verhalten in der näheren Vergangenheit dem Verhalten in der näheren Zukunft sehr ähnlich



5.20

Direct-Mapped Cache

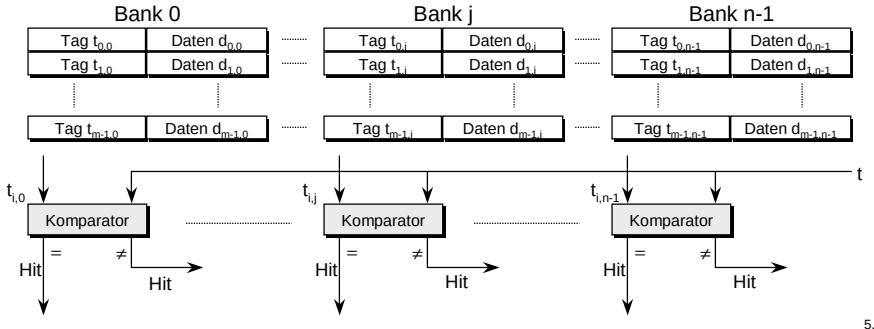
- Cache-Zeile wird direkt über die Adresse bestimmt
  - Tag wird als Adreßanteil in der Cache-Zeile gespeichert
  - Index bestimmt den Zeileneintrag
- Index beginnt bei LSB
  - Warum?



5.21

n-Wege-assoziativer Cache

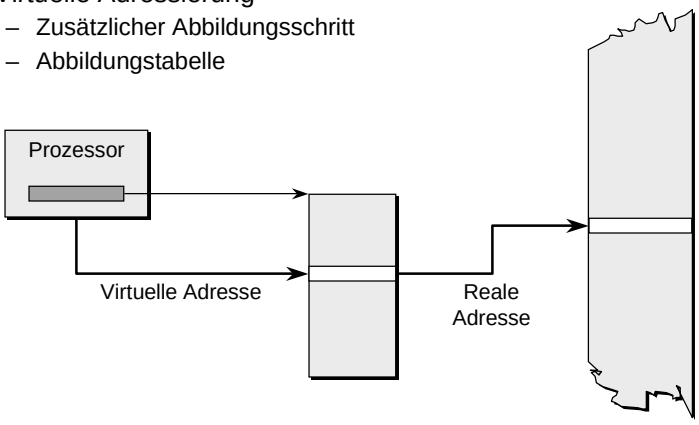
- "Halbassoziativer" Cache
  - Einsteig in den Cache analog zu Direct Mapped
  - n parallele Tabellen werden durchsucht
  - Zeilenersetzung auf n Cache-Zeilen
- n = 2 oder n = 4 gängig



5.22

Virtuelle Speicherverwaltung

- Reale Adressierung
  - Adresse referenziert Speicherzelle im physischen Speicher
- Virtuelle Adressierung
  - Zusätzlicher Abbildungsschritt
  - Abbildungstabelle



5.23

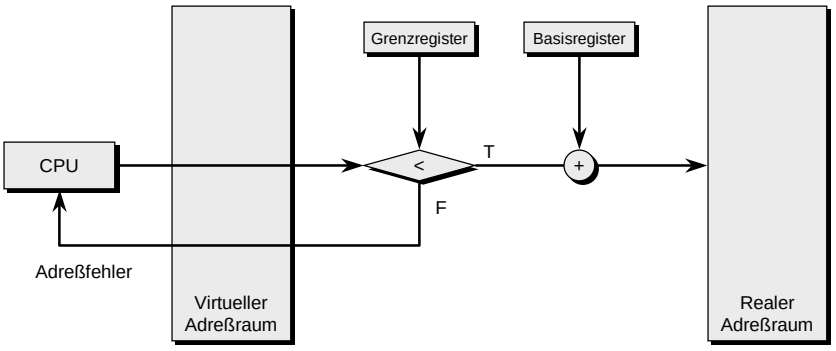
Vorteile?

5.24

Größe der Abbildungstabelle?

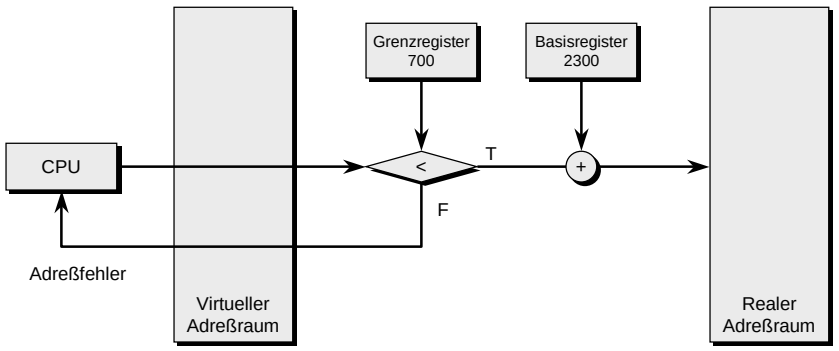
5.25

Virtuelle Adressierung: Triviallösung



5.26

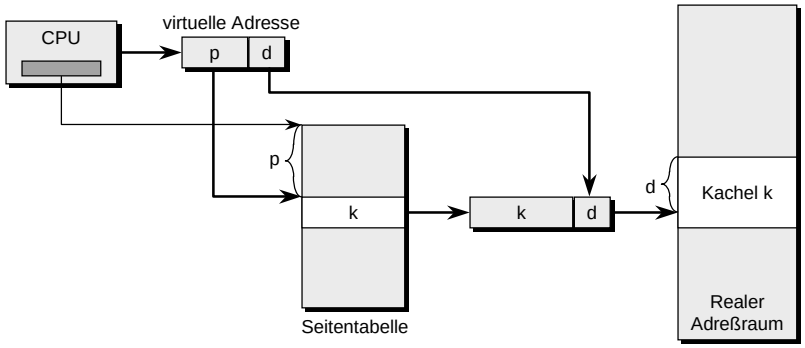
Beispiel



- Zugriff auf Adressen:
  - 0?
  - 200?
  - 699?
  - 700?
  - 1000?

5.27

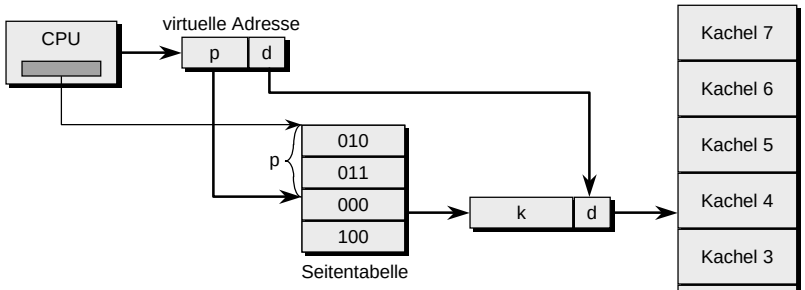
Seitenbasierte virtuelle Adressierung



- Unterteilung des Adreßraums in Seiten
- Einstufiges Abbildungsverfahren

5.28

Beispiel



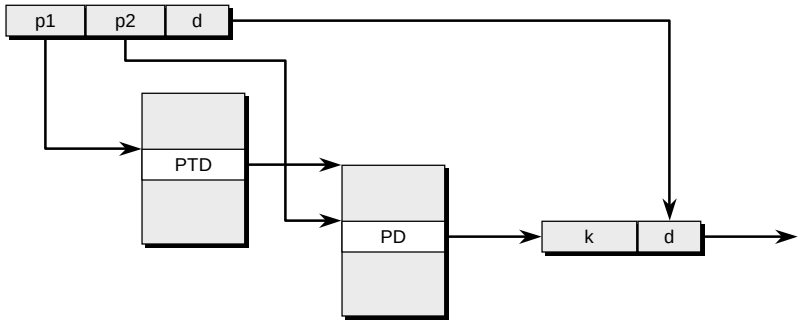
- Vorgaben:
  - 2 Bit Seitenadresse
  - 4 Bit Displacement
  - 3 Bit Kacheladresse

Größe der Seitentabelle

- 2^P Seiten zu je 2^D Speicherzellen:
  2^P Einträge in der Seitentabelle
- Größe eines Seitentabelleneintrags 4 - 6 Byte
- Beispiel (4 Byte pro Eintrag):
  - p = 2: 4 * 4 = 16 Byte
  - 32 Bit Adresse: 12 Bit Displacement (4KByte große Seiten)
    20 Bit Seitenadresse

    2^{20} * 4 Byte = 4 Mbyte große Seitentabelle
- Seitentabelle meist nur am Anfang und Ende belegt

Zwei- und mehrstufige Verfahren

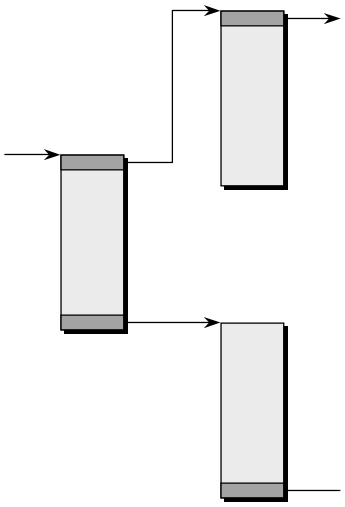


- z.B. 80386-Prozessor (p1 = p2 = 10 Bit Länge)
- PTD = Seitentabellen-Deskriptor
  - zeigt auf die nächst-tiefere Seitentabelle
- PD = Seiten-Deskriptor
  - zeit auf eine Kachel (falls aktuell eingeblendet)

5.31

Vorteil mehrstufiger Verfahren

- Nur die Wurzeltabelle muß initial angelegt sein
- Seitentabellen der tieferen Stufen nur bei Bedarf
  - z.B. Erstes und letztes Byte benutzt (p1=p2=10):
  - $3 * 2^{10} * 6$ Byte = 18 Kbyte



5.32

Beispiel

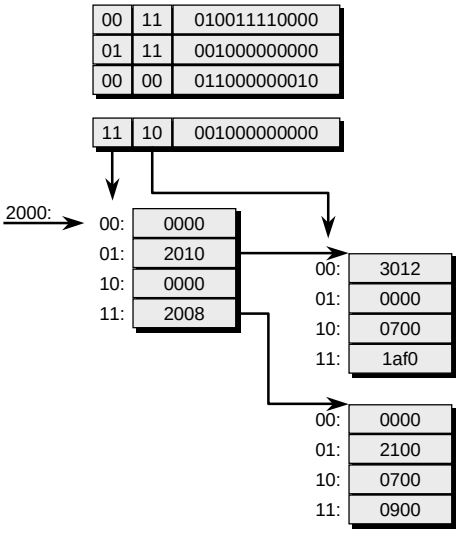
- Größen:
  - p1 = p2 = 2 Bit
  - d = 12 Bit
- Seitengröße?
- Wurzeltabelle bei 2000
- Virtuelle Adressen:
  - `34f0`?
  - `7200`?
  - `0602`?
  - `e200`?

- Speicherauszug:

```
2000: 0000
2002: 2010
2004: 0000
2006: 2008
2008: 0000
200a: 2100
200c: 0700
200e: 0900
2010: 3012
2012: 0000
2014: 0700
2016: 1af0
2018: 2020
```


5.33

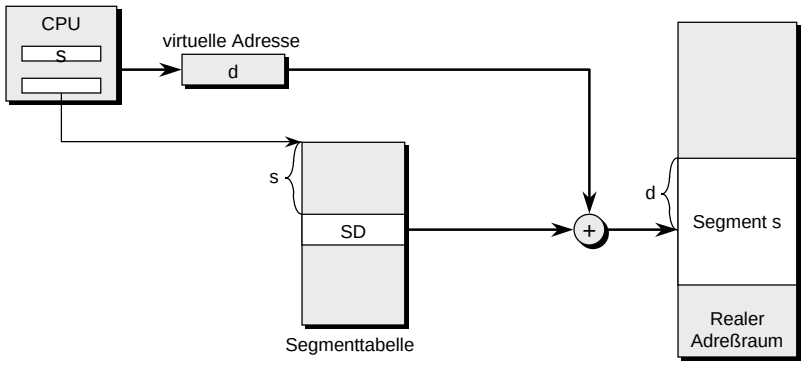
Lösung



- 12 Bit Displacement = 2^{12} Byte (4 Kbyte) Seitengröße
- Adreßrechnung:
  `34f0:` Adreßfehler
  `7200: 1af0+200 = 1af0200`
  `0602:` Adreßfehler
  `e200: 0700+200 = 0700200`

5.34

Segmentbasierte virtuelle Adressierung



- Segment-Deskriptor (SD):
  - Basisadresse (Segmentanfang)
  - Länge

5.35

Vergleich

Seitenbasierte Verfahren

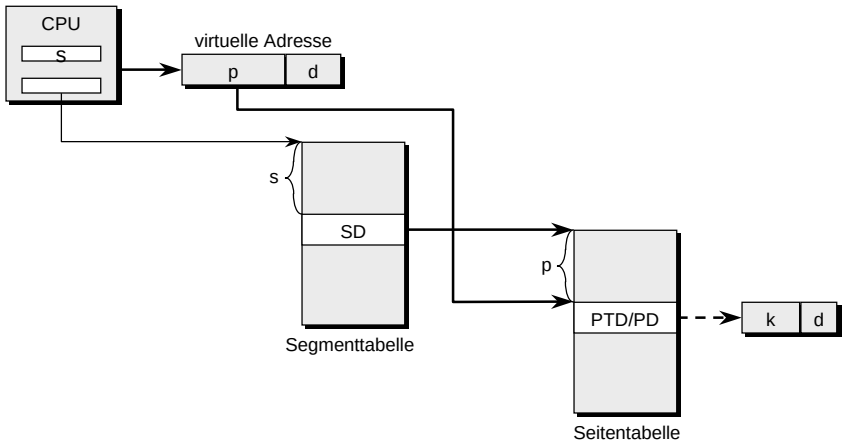
- Seiten fester Länge
  - Viele Seiten
  - Tabelle proportional zum Seitenbedarf
- Transparent für Anwendungen
- Adreßrechnung:
  - Abbildung

Segmentbasierte Verfahren

- Segmente variabler Länge
  - Wenig Segmente
  - Tabelle proportional zum Segmentbedarf
- Segmentstruktur für Anwendungen sichtbar
- Adreßrechnung:
  - Abbildung
  - Addition

5.36

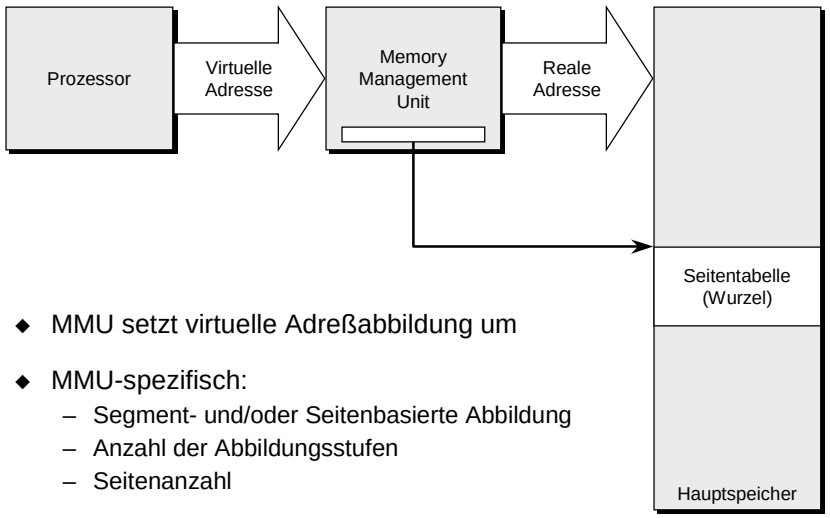
Segment- und Seitenbasierte Verfahren



- Segmente haben unterschiedliche Länge

5.37

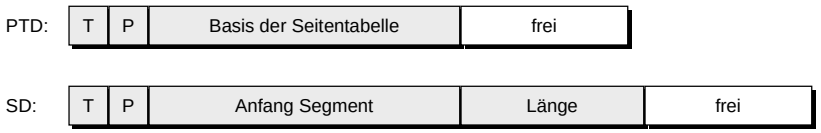
Die MMU



- MMU setzt virtuelle Adreßabbildung um
- MMU-spezifisch:
  - Segment- und/oder Seitenbasierte Abbildung
  - Anzahl der Abbildungsstufen
  - Seitenanzahl

5.38

Segment- und Seitentabellen-Deskriptoren



- T-Bit ermöglicht Unterscheidung:
  - Seitentabellen-Deskriptor (PTD)
  - Seiten-Deskriptor (PD)
  - Segment-Deskriptor (SD)
- P-Bit:
  - Seiten- oder Segmenttabelle im Hauptspeicher vorhanden (P=1)

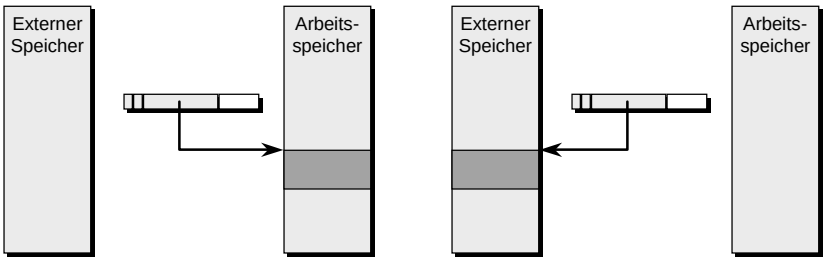
5.39

Seiten-Deskriptoren

- Schutzbits:
  - Erlaubte Zugriffsmodi (Lesen, Schreiben, Ausführen, ...)
  - Alle Bits 0 = keine Zugriffsform erlaubt
  - Schutzverletzung beim Zugriff bedeutet Fehler
- C-Bit (Cache Disable Bit)
  - Inhalte der Seite dürfen in Caches nicht zwischengespeichert werden
- D-Bit (Dirty-Bit)
  - Seit letztem Zurücksetzen fand ein Schreibzugriff statt
- R-Bit (Referenced-Bit)
  - Seit letztem Zurücksetzen wurde auf Seite in irgendeiner Form zugegriffen

5.40

P-Bit (Present-Bit)

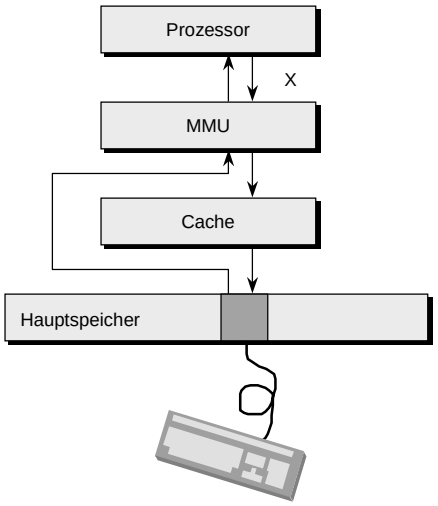


- P = 1:
  Seite im Arbeitsspeicher
- Speicherzellen direkt adressierbar

- P = 0:
  Seite ausgelagert
- Adreßfehler: Speicherzelle nicht direkt adressierbar
- Kein Arbeitsspeicher wird belegt

5.41

C-Bit (Cache Disable)

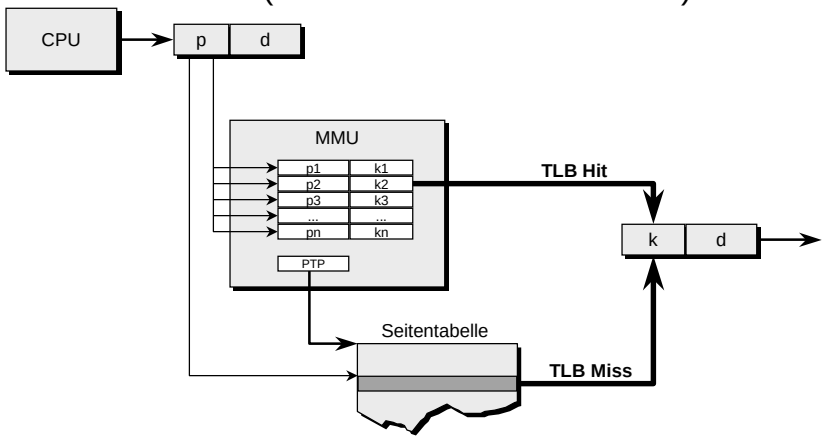



```
...
while (1) {
    c = getchar(keyboard);
    ...
    }
```

- Ohne C-Bit:
  - Lesezugriff auf Geräteregister (eingeblendet an Adresse X)
  - erster Tastendruck wird im Cache zwischengespeichert
  - alle nachfolgenden Leseoperationen von Adresse X sind Treffer

5.42

Der TLB (Translation Lookaside Buffer)



- TLB = Assoziative Suche nach Einträgen für virtuelle Seiten
- Voraussetzung für hohe Trefferquote: Lokalität

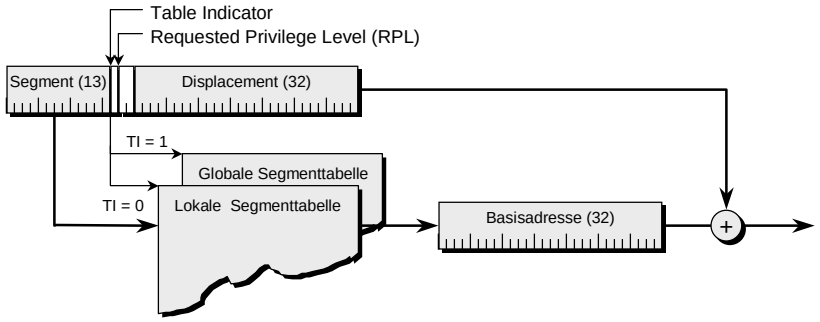
5.43

Die 80386 MMU

- On-Chip MMU
- Unterstützung für Segmente und Seiten (einzeln abschaltbar)
  - Real: 4 Gbyte linearer Adreßraum
  - Segmente: 64 Tbyte Adreßraum pro Prozeß
    - 2*8192 Segmente
    - max. 4 Gbyte pro Segment
  - Seiten: 4 Gbyte zweistufiger, seitenbasierter Adreßraum
    - 2^{10} Einträge jeweils in der ersten und zweiten Stufe
    - 4 Kbyte große Seitentabellen
    - 4 Kbyte Seitengröße
  - Segmente und Seiten: 64 Tbyte Adreßraum
    - 2*8192 Segmente
    - für jedes Segment zweistufige, seitenbasierte Adreßabbildung

5.44

80386: Segmentierung



- 2^{46} Bytes (=64 Tbyte) virtueller Adreßraum pro Prozess:
  - 32 Tbyte globales Segment (für alle Prozesse gleich, TI=1)
  - 32 Tbyte lokales Segment

5.45

80386: Segment-Deskriptor

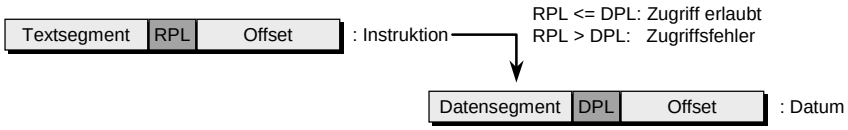
Segmentbasis 31..24	G	000	Länge 19..16	P	DPL	1	Typ	A	Segmentbasis 23..16
Segmentbasis 15..0				Segmentlänge 15..0					

- Maximale Segmentlänge: 20 Bit
  - Gesamtgröße abhängig vom Granularitätsbit (G)
  - G = 0: 2^{20} * 1 Byte = 1 MByte
  - G = 1: 2^{20} * 4 Kbyte = 4 Gbyte
- P = Präsenzbit
- DPL = Descriptior Privilege Level
- Typ zur Unterscheidung verschiedene Segmenttypen
  - u.a. Code- und Datensegment
  - legt Zugriffsrechte fest

5.46

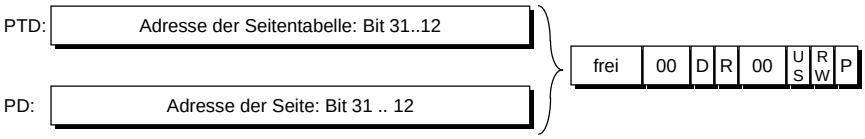
80386: Classification and Clearance

- DPL =
  - Classification bei Datensegment
  - Clearance bei Textsegment
- 4 Schutzebenen werden von Hardware unterstützt
- Verwendungsvorschlag:
  - 0 (Maximal): Interrupt-Routinen, Speicherverwaltung, Schutz
  - 1: Betriebssystem
  - 2: spezielle Server-Prozesse
  - 3 (Minimal): Anwendungsprogramme



5.47

80386: Seitentabellen- und Seitendeskriptor

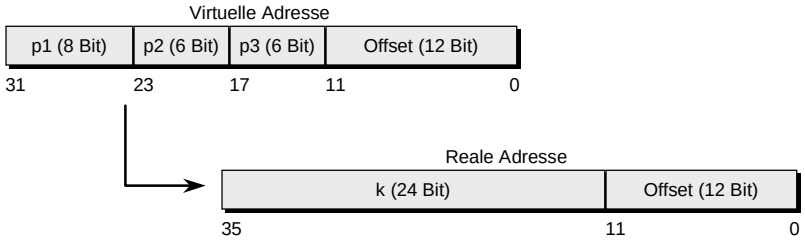


- 4 Byte große Deskriptoren
- 4 Kbyte große Seiten und Seitentabellen
- Seiten und Seitentabellen beginnen an einer 4 Kbyte-Grenze
- Bits:
  - Dirty-Bit (D)
  - Referenced-Bit (R)
  - User-Supervisor/Bit (US)
  - Read/Write/Bit (RW): nur im User-Modus
  - Present-Bit (P)

5.48

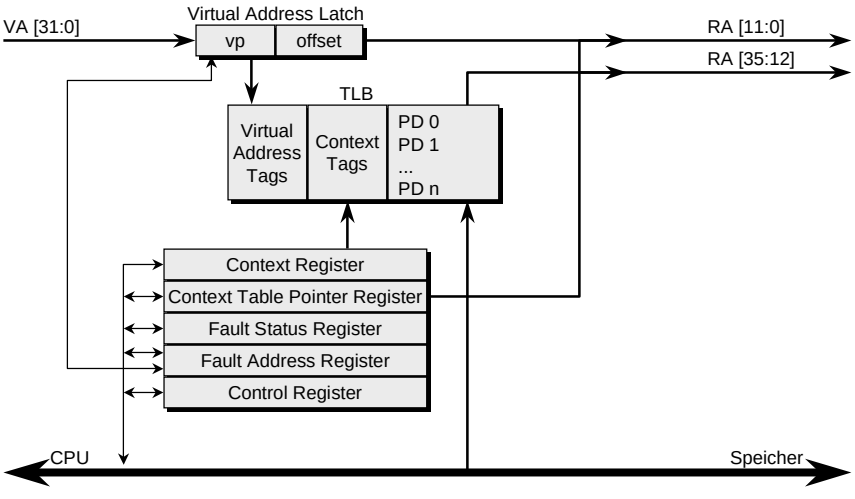
Die SPARC MMU

- Wesentliche Eigenschaften:
  - 32 Bit virtuelle Adressen
  - 36 Bit reale Adressen
  - 4 Kbyte Seitengröße
  - 3-stufige Adreßabbildung
  - Unterstützung mehrerer Kontexte



5.49

SPARC: Aufbau der MMU



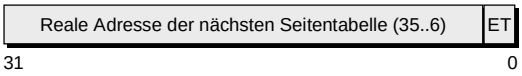
5.50

SPARC: Die MMU-Register

- Control Register
  - Implementierung und Version der MMU
  - Speichermodell
  - Art der Fehlerbehandlung
  - Ein- und Ausschalten der MMU
- Context Table Pointer Register
  - Zeigt auf die aktuelle Kontexttabelle
    - Zeiger auf die Ebene 1 Seitentabelle
    - Kontextkennung (32 Bit)
- Context Register
  - speichert die aktuell gültige Kontextkennung
- Fault Status und Fault Address Register
  - Fehlerart (Zugriffsverletzung, Kein Tabelleneintrag, ...)
  - fehlerhafte Adresse

5.51

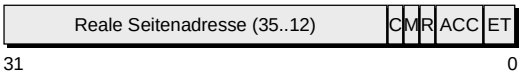
SPARC: Seitentabellen-Deskriptor



- Seitentabellen beginnen an einer 256 Byte Grenze
- Größe der Seitentabellen:
  - Ebene 1: 1024 Bytes (256 Einträge)
  - Ebene 2: 256 Bytes (64 Einträge)
  - Ebene 3: 256 Bytes (64 Einträge)
- ET = Entry Type:
  - 00: Ungültiger Eintrag (Adreßraum nicht angelegt)
  - 01: PTD
  - 10: PD
  - 11: Reserviert

5.52

SPARC: Seitendeskriptor

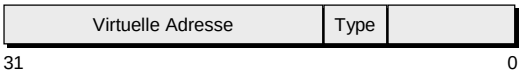


- C-Bit (Cache Disable)
- M-Bit (Modified Bit = Dirty Bit)
- R-Bit (Referenced Bit)
- ACC:

	User	Supervisor
000:	Read Only	Read Only
001:	Read / Write	Read / Write
010:	Read / Execute	Read / Execute
011:	Read / Write / Execute	Read / Write / Execute
100:	Execute Only	Execute Only
101:	Read Only	Read / Write
110:	-/-	Read / Execute
111:	-/-	Read / Write / Exec

5.53

SPARC: Flush and Probe



- Spezielle privilegierte Lese- und Schreibinstruktionen
- Flush: Bestimmte Einträge im TLB werden gelöscht
- Probe: CPU liest bestimmten TLB-Eintrag (oder Fehler)
- Type:

	Probe	Flush
000 (Seite)	Ebene 3 Eintrag	Ebene 3 PD
001 (Segment)	Ebene 2 Eintrag	Ebene 2-3 PTD/PD
010 (Region)	Ebene 1 Eintrag	Ebene 1-3 PTD/PD
011 (Context)	Ebene 0 Eintrag	Ebene 0-3 PTD/PD
100 (Entire)		Alle PTD/PD

5.54