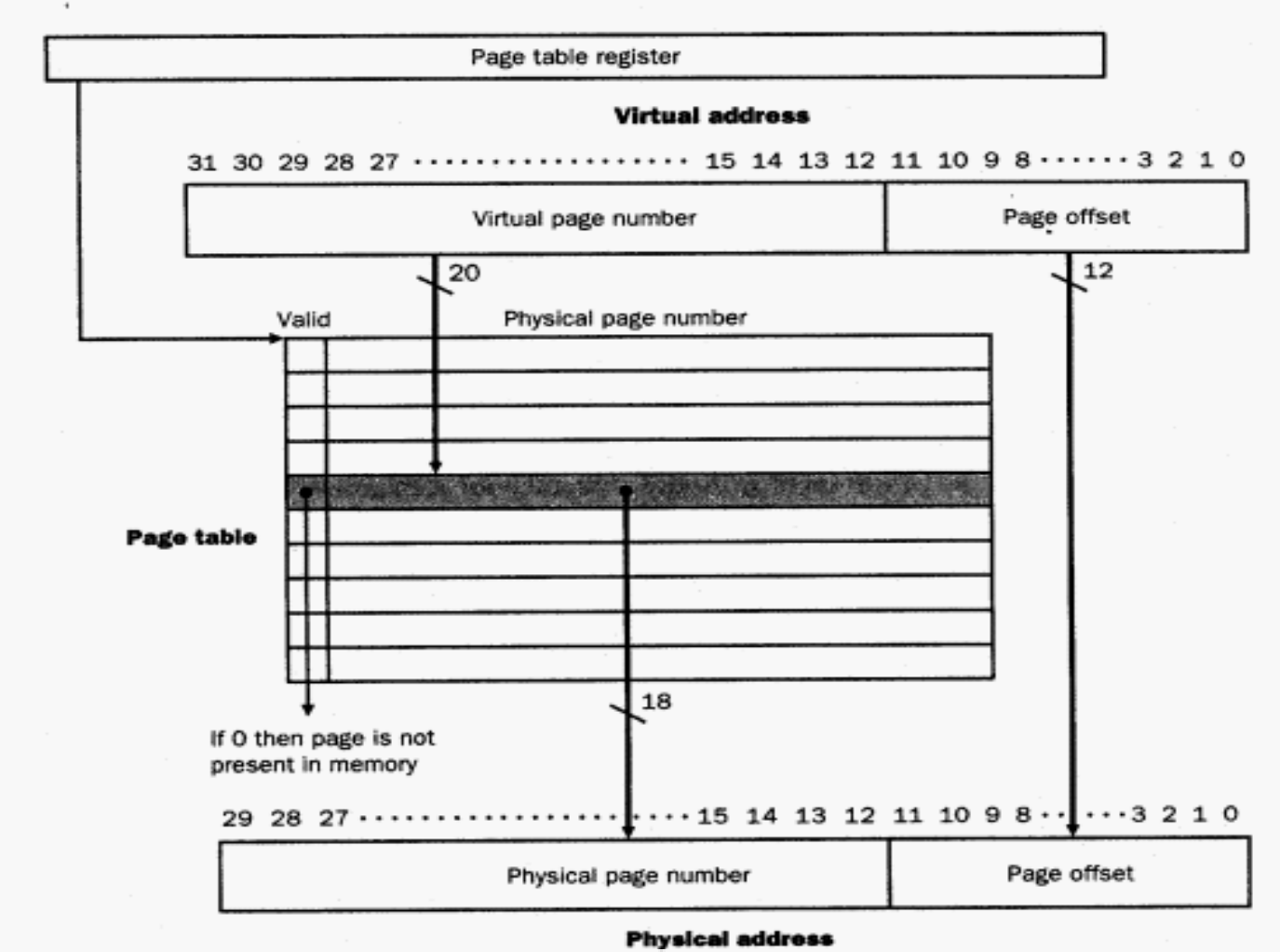
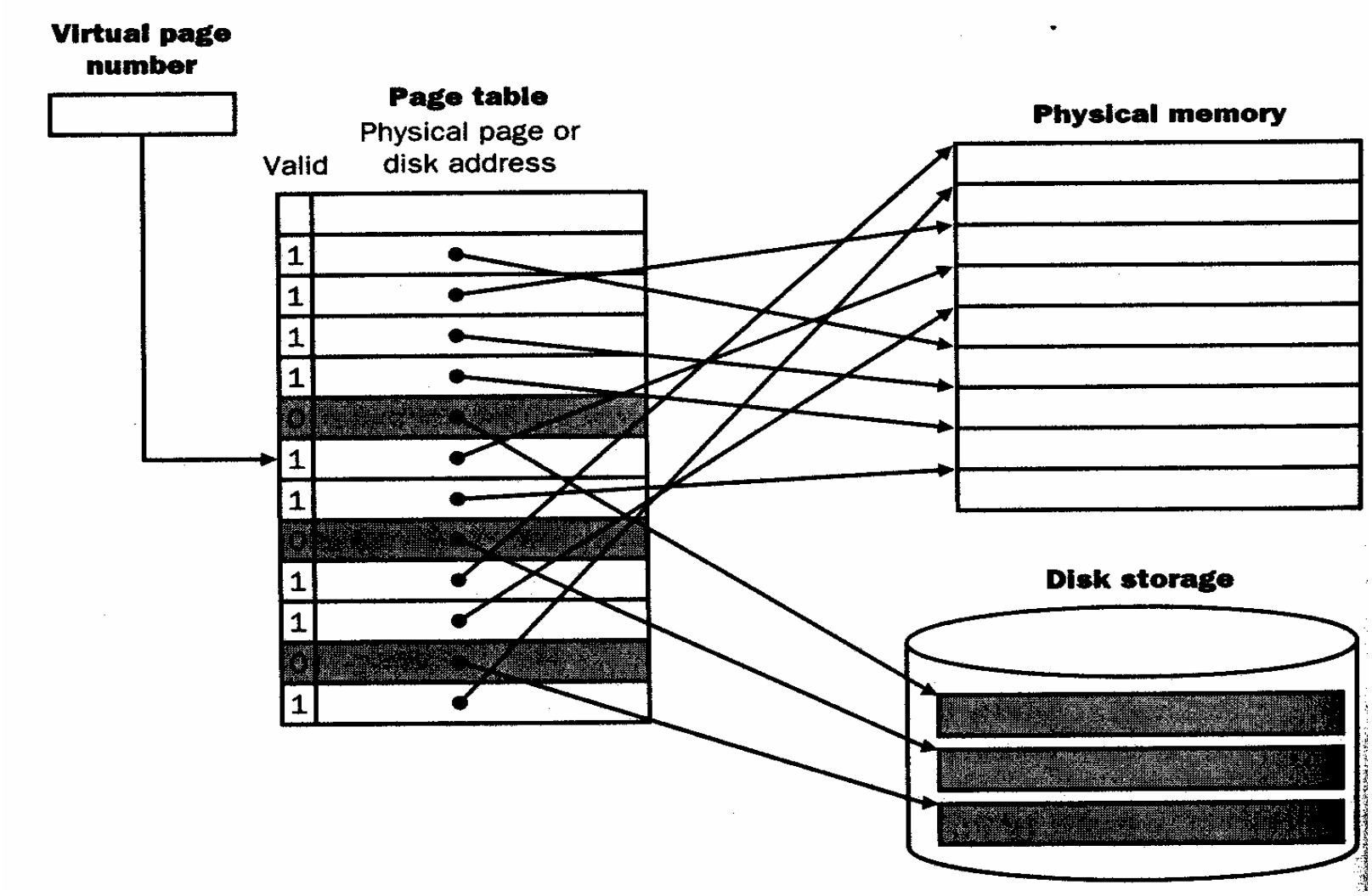
Speicherarchitektur (23)

Suchen einer Seite:



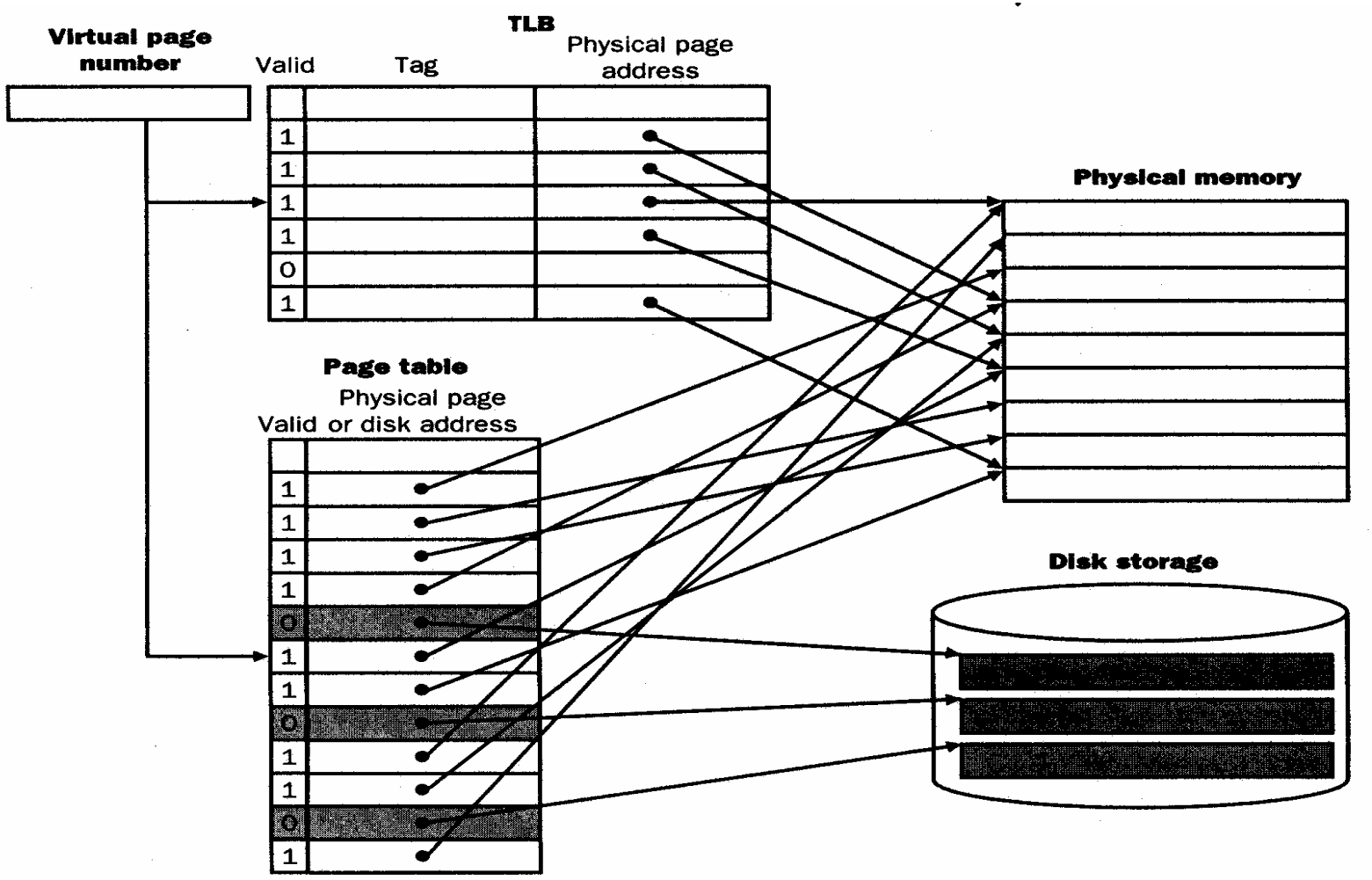
Speicherarchitektur (24)

Adressschema inklusive Seitenfehler:



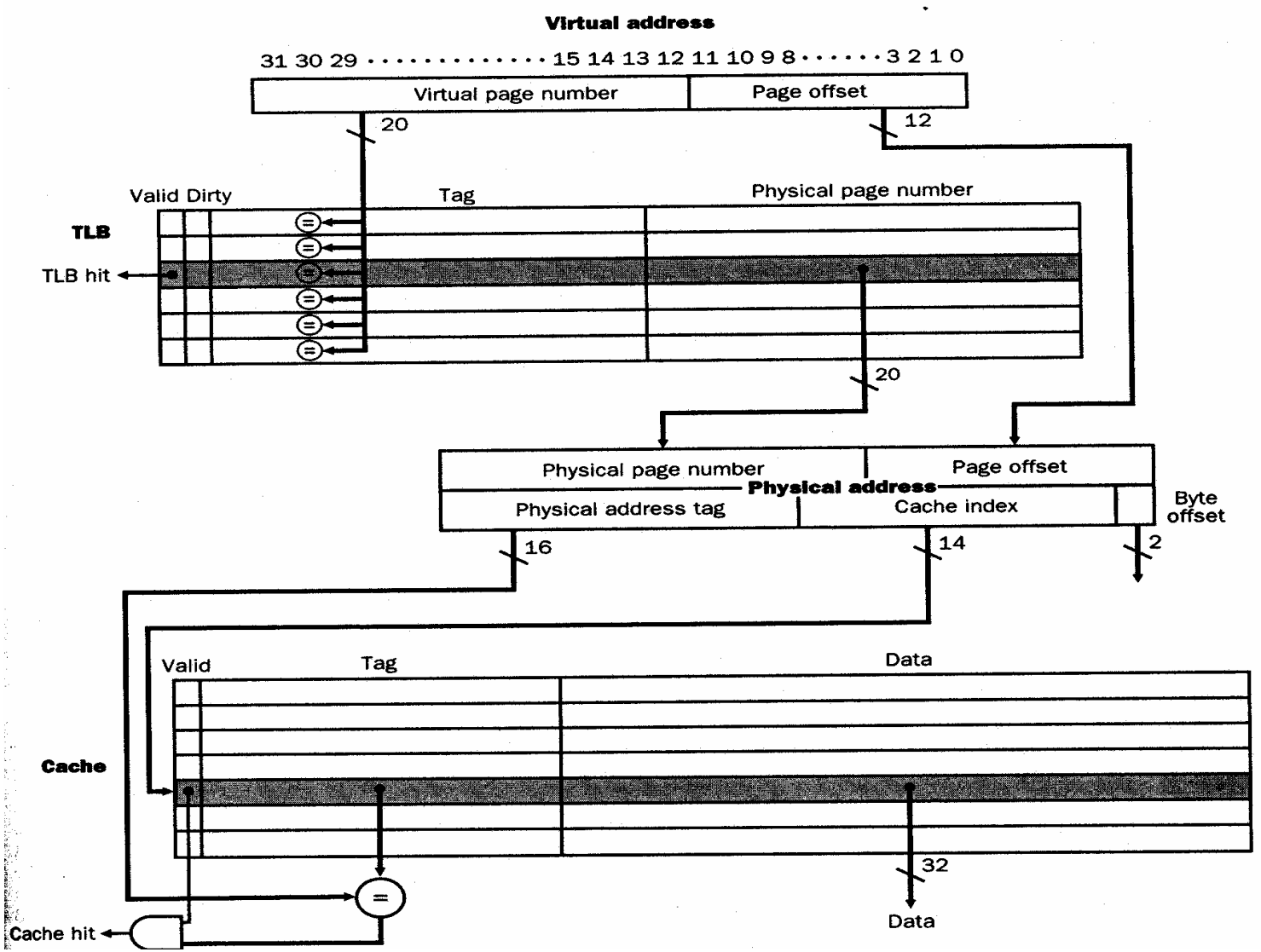
Speicherarchitektur (25)

TLB (Translation Lookaside Buffer):



Speicherarchitektur (26)

Der MIPS R2000 TLB und ein Cache in der DECStation 3100:



Speicherarchitektur (28)

Zusammenspiel von, Cache, TLB und physikalischem Speicher:

Cache	TLB	Virtual memory	Possible? If so, under what circumstance?
miss	hit	hit	Possible, although the page table is never really checked if TLB hits.
hit	miss	hit	TLB misses, but entry found in page table; after retry, data is found in cache.
miss	miss	hit	TLB misses, but entry found in page table; after retry, data misses in cache.
miss	miss	miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
miss	hit	miss	Impossible: cannot have a translation in TLB if page is not present in memory.
hit	hit	miss	Impossible: cannot have a translation in TLB if page is not present in memory.
hit	miss	miss	Impossible: data cannot be allowed in cache if the page is not in memory.

Speicherarchitektur (29)

Virtuelles Speicherkonzept erlaubt

1. geschütztes „sharing“ des physikalischen Speichers:

   Jeder Prozess hat seinen eigenen virtuellen Adressraum und das BS sorgt dafür, dass die Adressübersetzung zu jedem Zeitpunkt keine gemeinsamen Bilder enthält --->

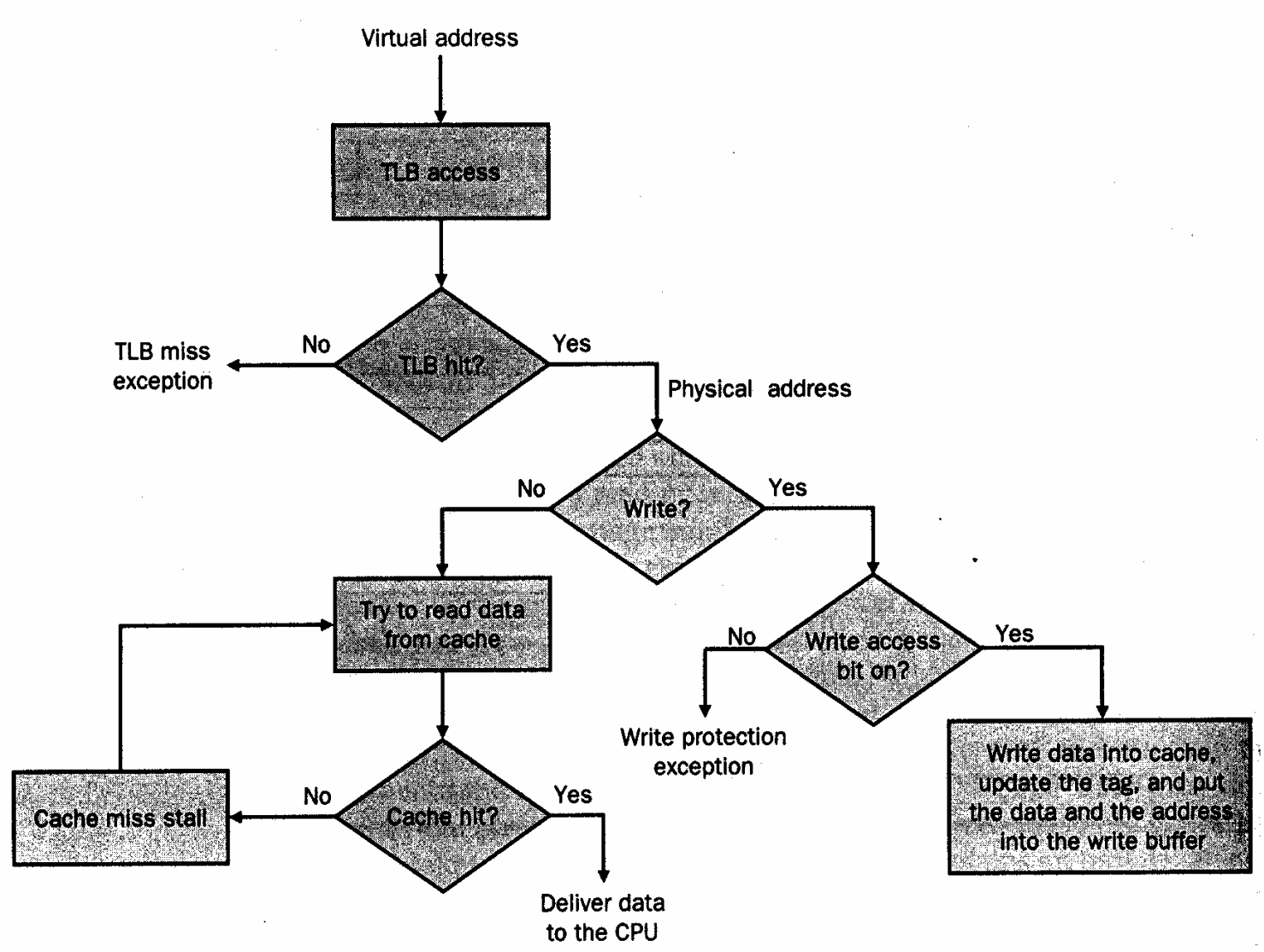
   - Nur das BS hat schreibenden Zugriff auf die Seitentabellen
   - Seitentabellen sind im Adressraum des BS plaziert

2. kontrolliertes „sharing“ von Seiten zwischen Prozessen:

   BS legt entsprechende Zugriffbits in den Seitentabellen an, die z.B. anzeigen, ob auf die entsprechende Seite lesend oder schreibend zugegriffen werden darf.

Speicherarchitektur (29a)

Bearbeitung eines Lese- bzw. Schreibbefehls:



Speicherarchitektur (31)

Zusammenfassung (Speicherhierarchie):

Quantitative Schlüsselentwurfsparameter:

Feature	Typical values for caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	1000–100,000	2000–250,000	32–4,000
Total size in kilobytes	8–8,000	8000–8,000,000	0.25–32
Block size in bytes	16–256	4000–64,000	4–32
Miss penalty in clocks	10–100	1,000,000–10,000,000	10–100
Miss rates	0.1%–10%	0.00001%–0.0001%	0.01%–2%

Diskussion der Hauptentwurfsalternativen

Frage 1: Wohin wird ein Block plaziert?

Scheme name	Number of sets	Blocks per set
Direct mapped	Number of blocks in cache	1
Set associative	$\frac{\text{Number of blocks in cache}}{\text{Associativity}}$	Associativity (typically 2–8)
Fully associative	1	Number of blocks in the cache

Speicherarchitektur (32)

Frage 2: Wie wird ein Block wiedergefunden?

Associativity	Location method	Comparisons required
Direct mapped	index	1
Set associative	index the set, search among elements	degree of associativity
Full	search all cache entries	size of the cache
	separate lookup table	0

Für Vollassoziativität im virtuellen Speicher spricht:

- Fehler sind extrem teuer
- Es können z. T. ausgeklügelte Ersetzungsstrategien eingesetzt werden
- Seitentabelle kommt ohne Zusatzhardware und Suchen aus
- Immer größere Seiten reduzieren die Größe der Seitentabelle

Speicherarchitektur (33)

Frage 3: Welcher Block soll im Fall eines Fehlers ersetzt werden?

2 hauptsächliche Strategien:

- Zufall
- LRU bzw. approximiertes LRU (vor allem im virtuellen Speicher)

Frage 4: Was geschieht beim Schreiben?

2 hauptsächliche Strategien:
- write-through
- write-back (copy-back)

Vorteile von write-through:
- Lesefehler können einfacher behandelt werden, da ein Block nicht zurück geschrieben werden muss
- Schreibpuffer verbessern Performanz

Vorteile von write-back:
- Das Schreiben einzelner Worte durch den Prozessor kann sich an der Akzeptanzrate des vorgeschalteten (nicht nachgeschalteten) Speichers orientieren
- Mehrfach-Writes innerhalb eines Blocks erfordern nur ein write zum nachgeschalteten Speicher
- Dirty bits verbessern Performanz von Lesefehlern

Speicherarchitektur (35)

Vergleich Speicherhierarchien Pentium Pro und PowerPC 604

Adressübersetzung und TLB hardware:

Characteristic	Intel Pentium Pro	PowerPC 604
Virtual address	32 bits	52 bits
Physical address	32 bits	32 bits
Page size	4 KB, 4 MB	4 KB, selectable, and 256 MB
TLB organization	A TLB for instructions and a TLB for data Both four-way set associative Pseudo-LRU replacement Instruction TLB: 32 entries Data TLB: 64 entries TLB misses handled in hardware	A TLB for instructions and a TLB for data Both two-way set associative LRU replacement Instruction TLB: 128 entries Data TLB: 128 entries TLB misses handled in hardware

First-level caches:

Characteristic	Intel Pentium Pro	PowerPC 604
Cache organization	Split instruction and data caches	Split instruction and data caches
Cache size	8 KB each for instructions/data	16 KB each for instructions/data
Cache associativity	Four-way set associative	Four-way set associative
Replacement	Approximated LRU replacement	LRU replacement
Block size	32 bytes	32 bytes
Write policy	Write-back	Write-back or write-through

Parallelrechner (1)

Motivation:

- Bedarf für immer leistungsfähigere Rechner (Wissenschaft, Industrieforschung, Entwicklung)
  - Simulation von komplexen physikalischen oder biochemischen Vorgängen
  - Entwurfsunterstützung automotive und pharmazeutische Produkte
  - virtuelle Realität

- Leistungssteigerung eines einzelnen Rechners hat physikalisch/technologische Grenzen:
  – Geschwindigkeit von Materie
  – Wärmeableitung
  – Transistorgröße

„Bottom-up“ - Sicht:

Ansammlung von untereinander verdrahteten Chips (Prozessor- bzw. Speicherelemente) (Anzahl, Größe von jedem Typ? Wie sind die Elemente untereinander verbunden?)

„Top-down“ - Sicht:

Ansammlung von parallel auszuführenden Programm(teil)en (Anzahl, Größe?) mit welcher Art von Kommunikation?

Parallelrechner (2)

Themen:

- Kommunikationsmodelle (speicherbasiert/implizit, nachrichtenbasiert/explizit)
- Verbindungsnetzwerke (wenn nachrichtenbasiert)
- Metriken und Methoden zur Geschwindigkeitssteigerung durch Parallelität
- Taxonomie von Parallelrechnern

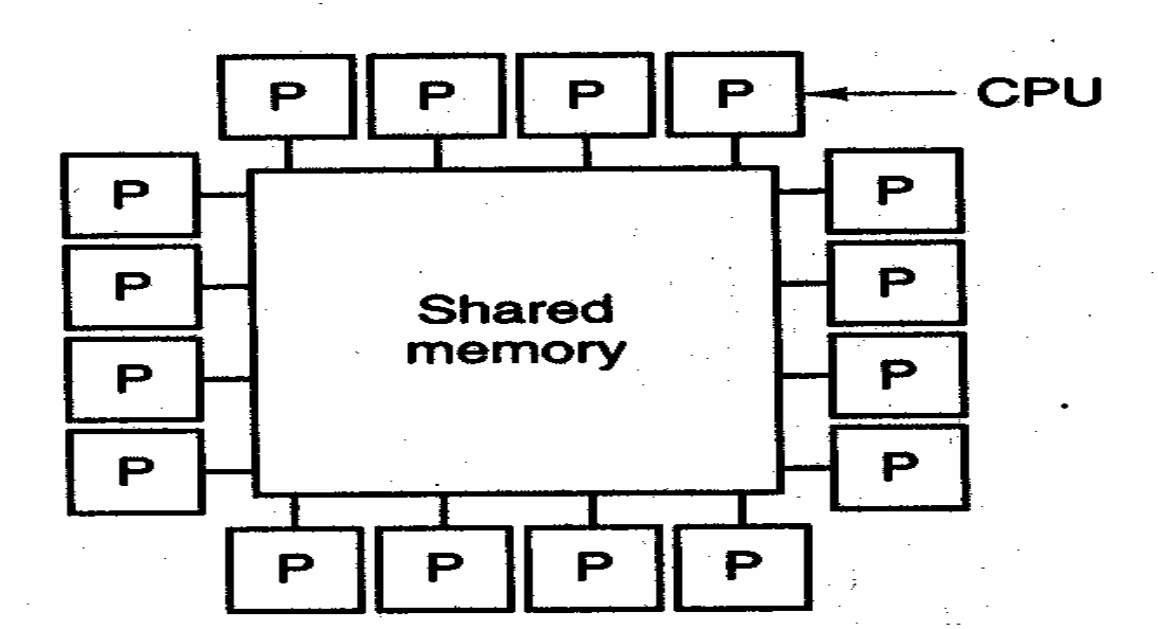
Parallelrechner (3)

Kommunikationsmodelle

Wie kommunizieren die parallel von mehreren CPU´s bearbeiteten Programmteile miteinander?

Multiprozessoren (gemeinsam benutzter Speicher)
Alle CPU´s kommunizieren über einen gemeinsam benutzten Speicher durch Ausführung entsprechender Schreib/Lese - Befehle

Multiprozessorsystem mit 16 CPU´s:

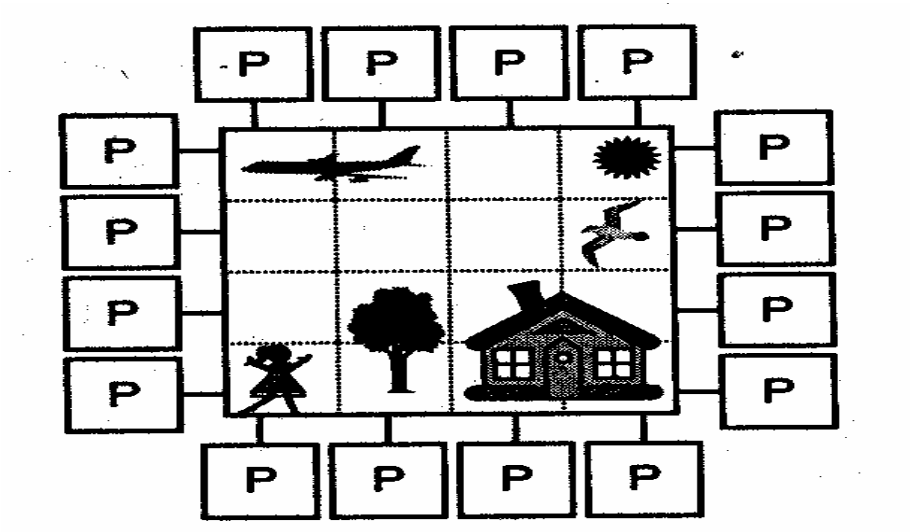


Hauptvorteil:

- einfaches Programmiermodell

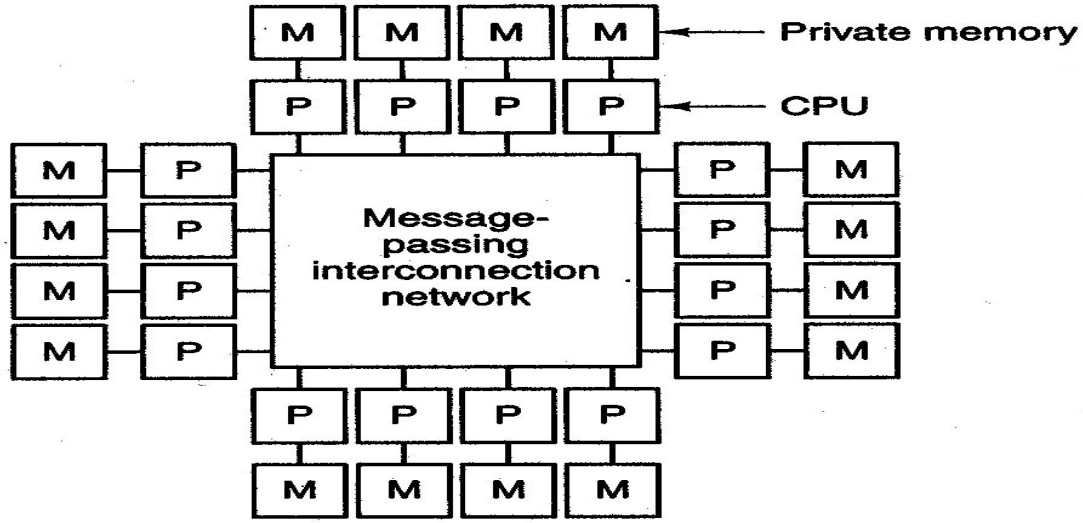
Parallelrechner (4)

Beispiel: Parallelanalyse eines Bildes



Multicomputer (verteilter Speicher)
Jede CPU hat ihren privaten Speicher. Kommuniziert wird durch über ein Verbindungsnetzwerk verschickte Nachrichten (Botschaften).

Multicomputersystem mit 16 CPU´s:

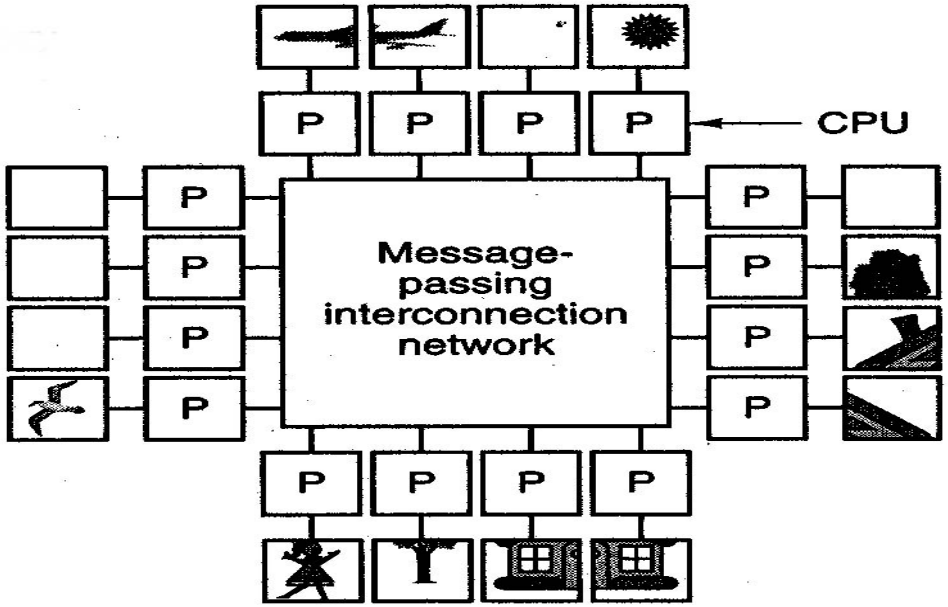


Parallelrechner (5)

Hauptvorteil:

- einfach zu entwerfen, billiger, skalieren besser

Beispiel: Parallelanalyse



Problem:

Die Vorteile beider Ansätze schließen sich gegenseitig aus:

Multiprozessoren sind schwierig zu entwerfen aber leicht zu programmieren
Multicomputer sind leicht zu entwerfen aber schwierig zu programmieren

---> Hybridsysteme