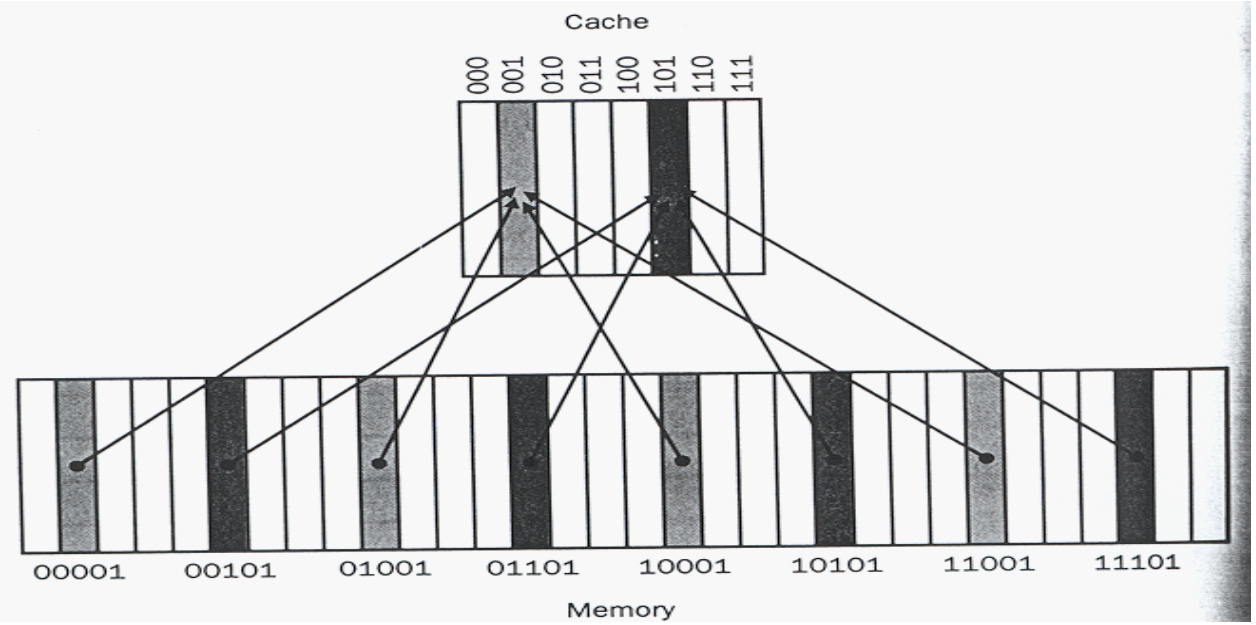
Speicherarchitektur (6)

Beispiel:



Beispiel für eine Folge von Prozessorreferenzierungen:

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss (7.5b)	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss (7.5c)	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss (7.5d)	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss (7.5e)	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss (7.5f)	$(10010_{two} \bmod 8) = 010_{two}$

Speicherarchitektur (7)

Entsprechende Folge von Cachebelegungen:

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

a. The initial state of the cache after power-on

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory(10110_{two})
111	N		

b. After handling a miss of address (10110_{two})

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

c. After handling a miss of address (11010_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

d. After handling a miss of address (10000_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

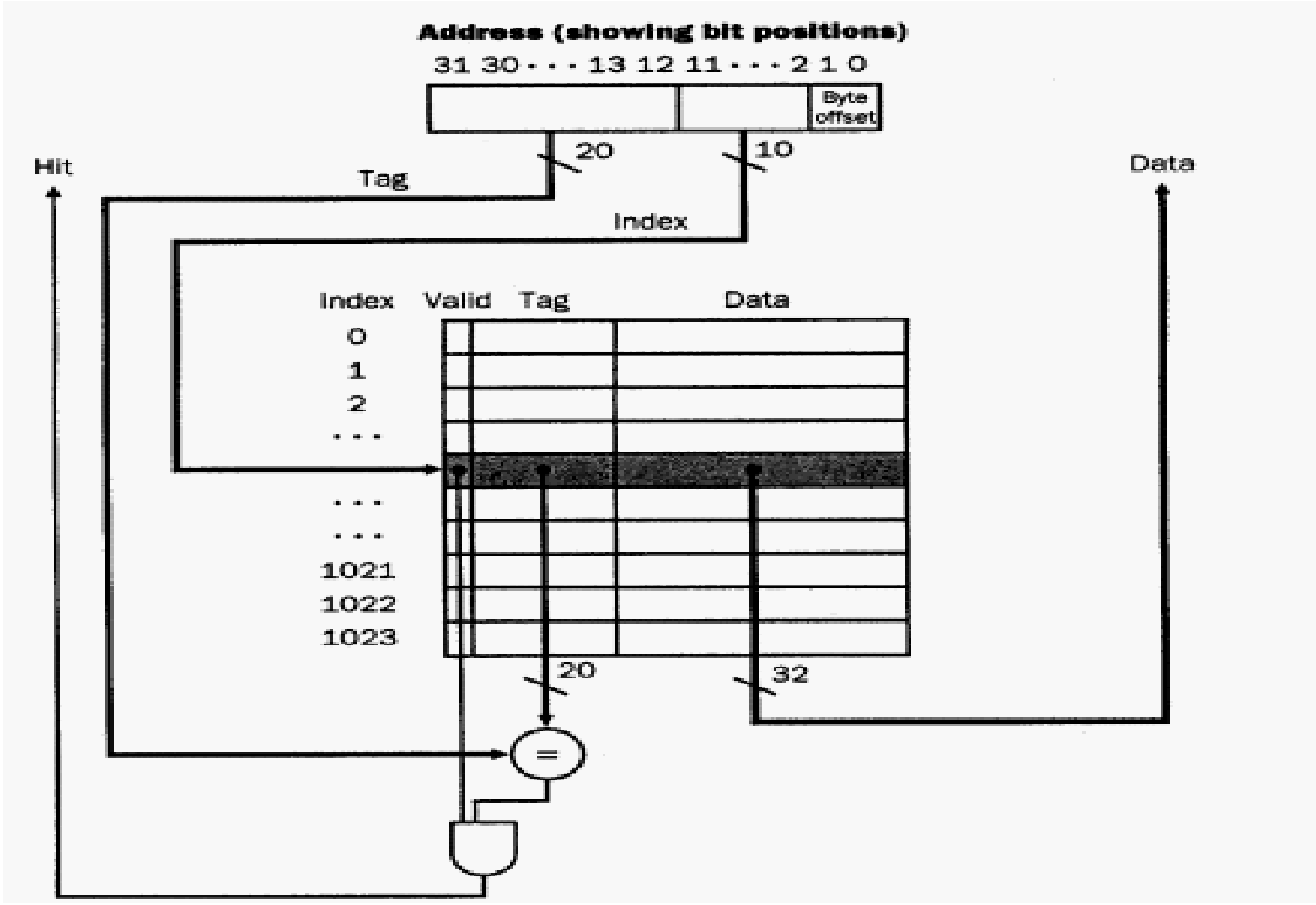
e. After handling a miss of address (00011_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

f. After handling a miss of address (10010_{two})

Speicherarchitektur (8)

Struktur einer referenzierten Adresse:



Speicherarchitektur (9)

Cache misses:

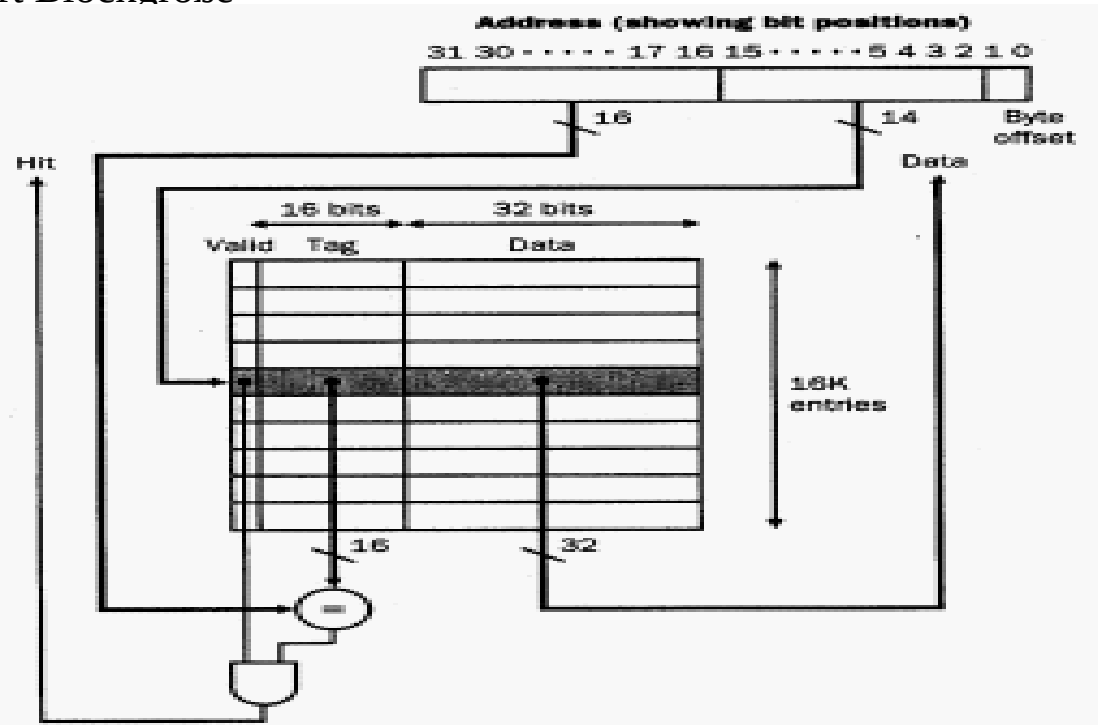
- Sende den PC-Wert zum (Haupt-)Speicher
- Mache entsprechenden Speicherzugriff
- Schreibe den entsprechenden Cacheeintrag
- Neustart der Befehlsausführung

Beispielcache: DECStation 3100

- separater Befehls- und Datencache (zur Unterstützung von pipelining)
- Cachegröße jeweils 16K Worte mit 1-Wort Blockgröße

  ---> index = 14 bits ---> tag = 16 bits

Struktur der referenzierten Adresse:



Speicherarchitektur (10)

Schreiben in Cache (write-through):

- Indiziere den Cache mit den Bitadressen von 15 bis 2
- Schreibe Bitadressen von 31 bis 16 in den tag, Datum in den Datenteil des Cache, setze das Gültigkeitsbit
- Schreibe Datum in Hauptspeicher unter Nutzung der Gesamtadresse

Vorteil von write-through: einfacher Mechanismus, da keine Unterscheidung zwischen Treffer und Fehler

Nachteil von write-through: schlechte Performanz

Alternativen:

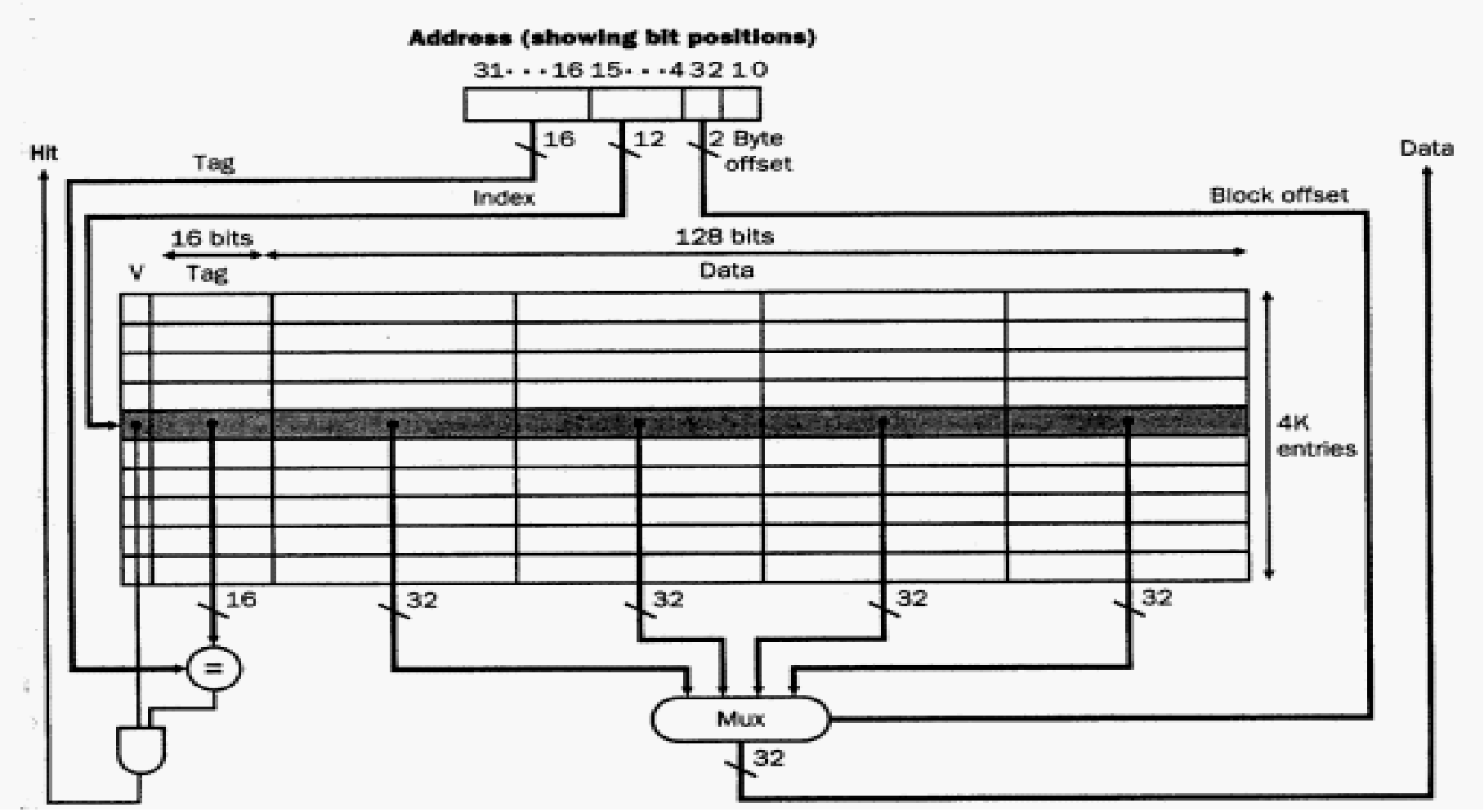
- Nutzung eines Schreibpuffers
- write-back

gemessene miss-Raten für DECStation 3100:

Program	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	6.1%	2.1%	5.4%
spice	1.2%	1.3%	1.2%

Speicherarchitektur (11)

Cache mit Blockgröße 4 Worte:

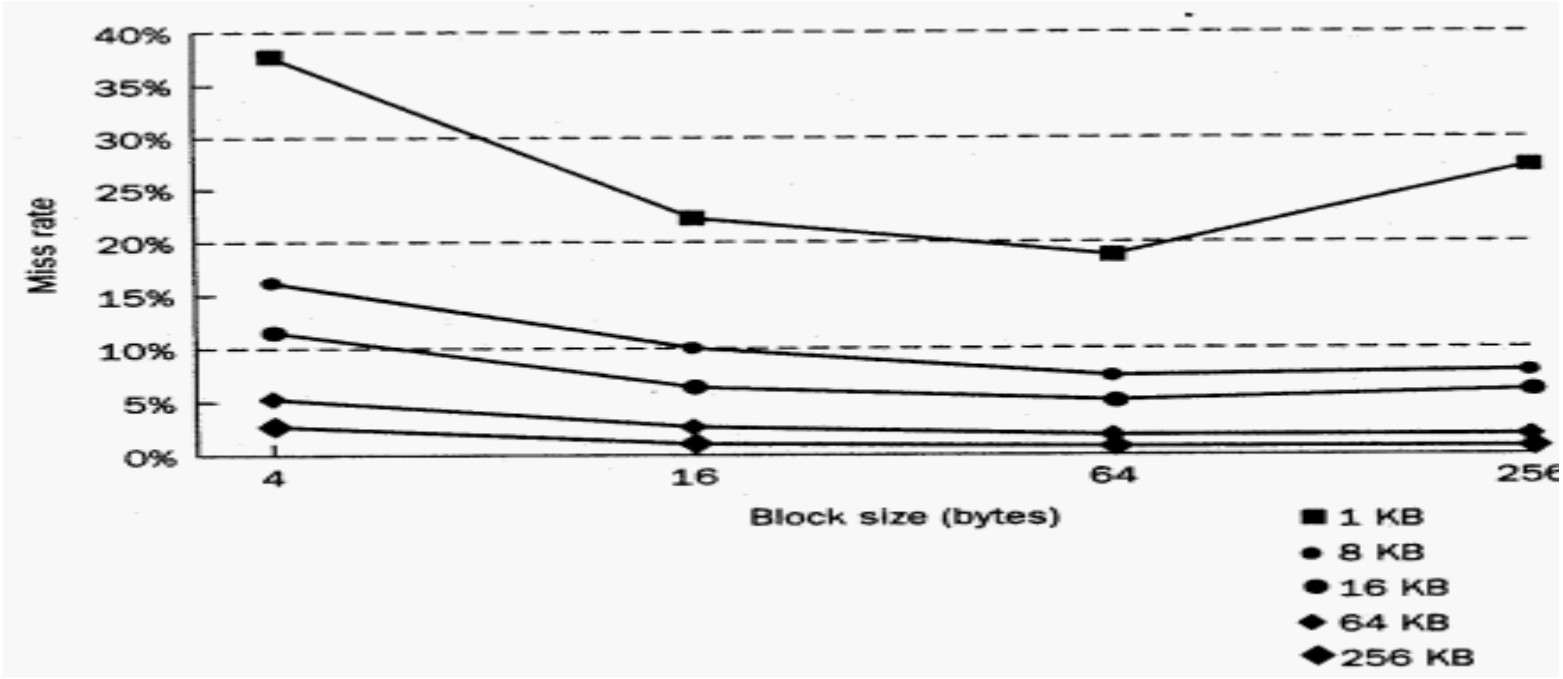


Speicherarchitektur (12)

miss-Rate für unterschiedliche Blockgrößen:

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

miss-Rate in Abhängigkeit von Blockgröße:



Speicherarchitektur (13)

Ergebnis: im Normalfall bewirkt Erhöhung der Blockgröße eine Reduzierung der miss-Rate
Ausnahme: Cache ist im Verhältnis zur Blockgröße zu klein

miss-penalty:= Kosten eines miss ausgedrückt in Zeit
Blockholzeit von nächster Ebene (= Zugriffszeit zum 1. Wort + Blocktransferzeit) + Cacheladezeit

weiteres Problem mit der Blockgröße:
wachsende Blockgröße ---> - steigende Transferzeit ---> höhere miss-penalty
- bei großen Blockgrößen wird der positive Effekt auf miss-Rate kleiner
---> cache performance insgesamt sinkt

Lösungsansätze:

- early restart:

Wiederaufnahme der Ausführung sobald das gesuchte Wort im Block vorhanden

  - eher geeignet für Befehlscache(meist sequentieller Zugriff)
  - weniger effektiv für Datencache auf Grund geringerer Lokalität, evtl. sogar kontraproduktiv

- critical word first

das gesuchte Wort im gesamten Block wird zuerst transferiert

  - leicht schneller als early restart
  - gleiche Beschränkungen wie bei early restart

- **Änderung des Speicherarchitekturentwurfs hin zu größeren Bandbreiten**

Speicherarchitektur (14)

3 Alternativen für den Entwurf: