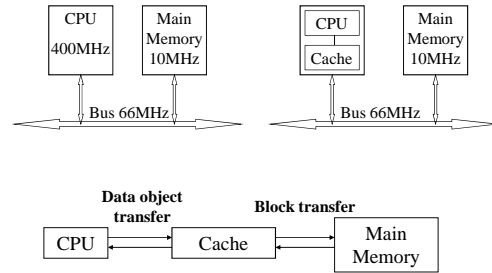


Memory Hierarchy Design

Chapter 5 and Appendix C

1

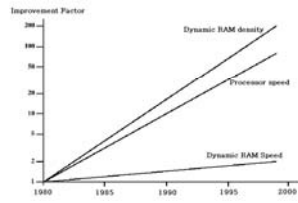
Cache Systems



4

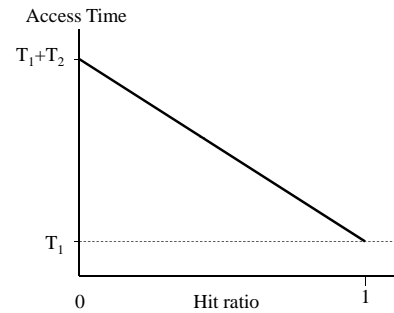
Overview

- Problem
 - CPU vs Memory performance imbalance
- Solution
 - Driven by temporal and spatial locality
 - Memory hierarchies
 - Fast L1, L2, L3 caches
 - Larger but slower memories
 - Even larger but even slower secondary storage
 - Keep most of the action in the higher levels



2

Example: Two-level Hierarchy



5

Locality of Reference

- Temporal and Spatial
- Sequential access to memory
- Unit-stride loop (cache lines = 256 bits)

```
for (i = 1; i < 100000; i++)
    sum = sum + a[i];
```
- Non-unit stride loop (cache lines = 256 bits)

```
for (i = 0; i <= 100000; i = i+8)
    sum = sum + a[i];
```

3

Basic Cache Read Operation

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

6

Elements of Cache Design

- Cache size
- Line (block) size
- Number of caches
- Mapping function
 - Block placement
 - Block identification
- Replacement Algorithm
- Write Policy

7

Number of Caches

- Increased logic density => on-chip cache
 - Internal cache: level 1 (L1)
 - External cache: level 2 (L2)
- Unified cache
 - Balances the load between instruction and data fetches
 - Only one cache needs to be designed / implemented
- Split caches (data and instruction)
 - Pipelined, parallel architectures

10

Cache Size

- Cache size \ll main memory size
- Small enough
 - Minimize cost
 - Speed up access (less gates to address the cache)
 - Keep cache on chip
- Large enough
 - Minimize average access time
- Optimum size depends on the workload
- Practical size?

8

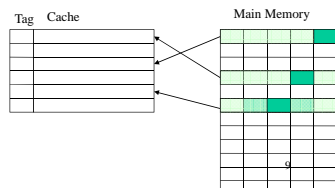
Mapping Function

- Cache lines \ll main memory blocks
- Direct mapping
 - Maps each block into only one possible line
 - (block address) MOD (number of lines)
- Fully associative
 - Block can be placed anywhere in the cache
- Set associative
 - Block can be placed in a restricted set of lines
 - (block address) MOD (number of sets in cache)

11

Line Size

- Optimum size depends on workload
- Small blocks do not use *locality of reference* principle
- Larger blocks reduce the number of blocks
 - Replacement overhead
- Practical sizes?



Cache Addressing

Block address		Block offset
Tag	Index	

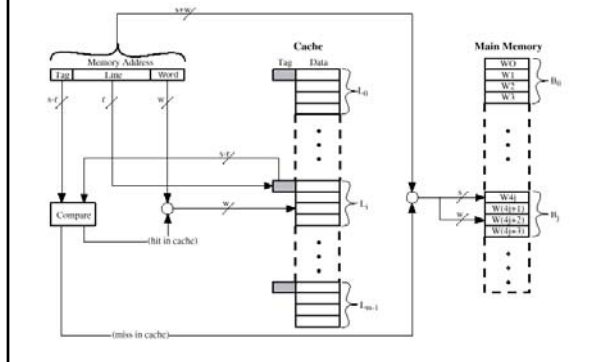
Block offset – selects data object from the block

Index – selects the block set

Tag – used to detect a hit

12

Direct Mapping



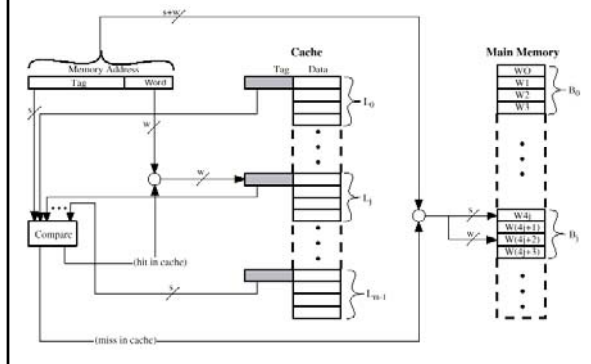
Replacement Algorithm

- Simple for direct-mapped: no choice
- Random
 - Simple to build in hardware
- LRU

Size	Associativity					
	Two-way		Four-way		Eight-way	
	LRU	Random	LRU	Random	LRU	Random
16KB	5.18%	5.69%	4.67%	5.29%	4.39%	4.96%
64KB	1.88%	2.01%	1.54%	1.66%	1.39%	1.53%
256KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

16

Associative Mapping

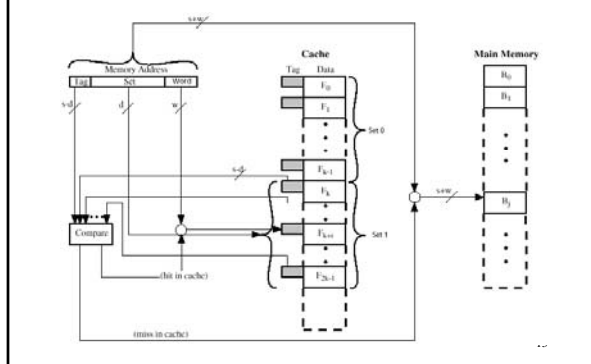


Write Policy

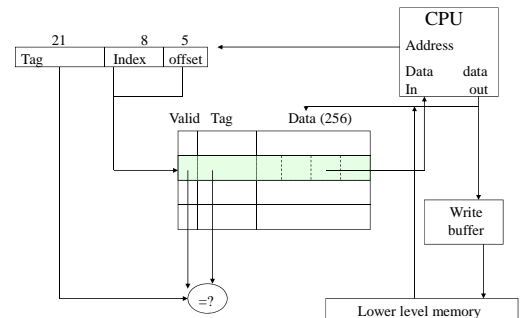
- Write is more complex than read
 - Write and tag comparison can not proceed simultaneously
 - Only a portion of the line has to be updated
- Write policies
 - Write through – write to the cache and memory
 - Write back – write only to the cache (dirty bit)
- Write miss:
 - Write allocate – load block on a write miss
 - No-write allocate – update directly in memory

17

K-Way Set Associative Mapping



Alpha AXP 21064 Cache



18

Write Merging

Write address	V	V	V	V
100	1	0	0	0
104	1	0	0	0
108	1	0	0	0
112	1	0	0	0

Write address	V	V	V	V
100	1	1	1	1
	0	0	0	0
	0	0	0	0
	0	0	0	0

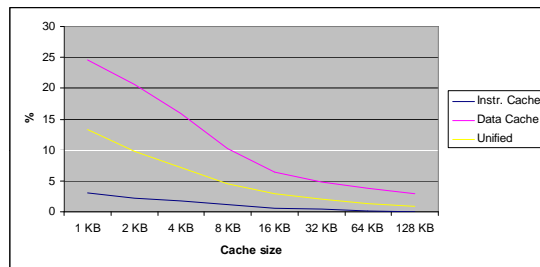
19

Cache Performance Improvements

- Average memory-access time
= Hit time + Miss rate x Miss penalty
- Cache optimizations
 - Reducing the miss rate
 - Reducing the miss penalty
 - Reducing the hit time

22

DECstation 5000 Miss Rates



Direct-mapped cache with 32-byte blocks

Percentage of instruction references is 75%

20

Example

Which has the lower average memory access time:

A 16-KB instruction cache with a 16-KB data cache or

A 32-KB unified cache

Hit time = 1 cycle

Miss penalty = 50 cycles

Load/store hit = 2 cycles on a unified cache

Given: 75% of memory accesses are instruction references.

Overall miss rate for split caches = $0.75 * 0.64\% + 0.25 * 6.47\% = 2.10\%$

Miss rate for unified cache = 1.99%

Average memory access times:

Split = $0.75 * (1 + 0.0064 * 50) + 0.25 * (1 + 0.0647 * 50) = 2.05$

Unified = $0.75 * (1 + 0.0199 * 50) + 0.25 * (2 + 0.0199 * 50) = 2.24$

23

Cache Performance Measures

- **Hit rate:** fraction found in that level
 - So high that usually talk about *Miss rate*
 - Miss rate fallacy: as MIPS to CPU performance,
- Average memory-access time
 - Hit time + Miss rate x Miss penalty (ns)
- **Miss penalty:** time to replace a block from lower level, including time to replace in CPU
 - *access time* to lower level = f(latency to lower level)
 - *transfer time:* time to transfer block = f(bandwidth)

21

Cache Performance Equations

$$CPU_{time} = (CPU \text{ execution cycles} + Mem \text{ stall cycles}) * Cycle \text{ time}$$

$$Mem \text{ stall cycles} = Mem \text{ accesses} * Miss \text{ rate} * Miss \text{ penalty}$$

$$CPU_{time} = IC * (CPI_{execution} + Mem \text{ accesses per instr} * Miss \text{ rate} * Miss \text{ penalty}) * Cycle \text{ time}$$

$$Misses \text{ per instr} = Mem \text{ accesses per instr} * Miss \text{ rate}$$

$$CPU_{time} = IC * (CPI_{execution} + Misses \text{ per instr} * Miss \text{ penalty}) * Cycle \text{ time}$$

24

Reducing Miss Penalty

- Multi-level Caches
- Critical Word First and Early Restart
- Priority to Read Misses over Writes
- Merging Write Buffers
- Victim Caches

25

Critical Word First and Early Restart

- Critical Word First: Request the missed word first from memory
- Early Restart: Fetch in normal order, but as soon as the requested word arrives, send it to CPU

28

Multi-Level Caches

- Avg mem access time = Hit time(L1) + Miss Rate (L1) X Miss Penalty(L1)
- Miss Penalty (L1) = Hit Time (L2) + Miss Rate (L2) X Miss Penalty (L2)
- Avg mem access time = Hit Time (L1) + Miss Rate (L1) X (Hit Time (L2) + Miss Rate (L2) X Miss Penalty (L2))
- Local Miss Rate: number of misses in a cache divided by the total number of accesses to the cache
- Global Miss Rate: number of misses in a cache divided by the total number of memory accesses generated by the cache

26

Giving Priority to Read Misses over Writes

SW R3, 512(R0)

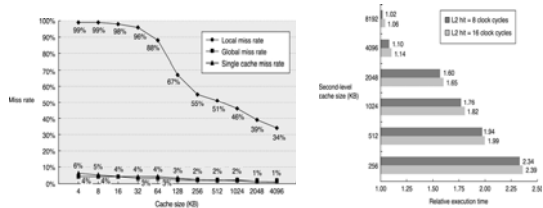
LW R1, 1024 (R0)

LW R2, 512 (R0)

- Direct-mapped, write-through cache mapping 512 and 1024 to the same block and a four word write buffer
- Will R2=R3?
- Priority for Read Miss?

29

Performance of Multi-Level Caches

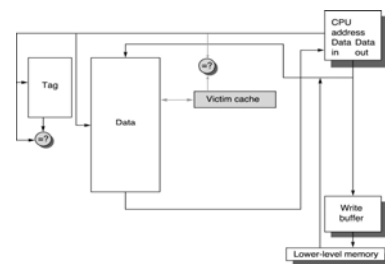


© 2003 Elsevier Science (USA). All rights reserved.

© 2003 Elsevier Science (USA). All rights reserved.

27

Victim Caches



© 2003 Elsevier Science (USA). All rights reserved.

30

Reducing Miss Rates: Types of Cache Misses

- Compulsory
 - First reference or cold start misses
- Capacity
 - Working set is too big for the cache
 - Fully associative caches
- Conflict (collision)
 - Many blocks map to the same block frame (line)
 - Affects
 - Set associative caches
 - Direct mapped caches

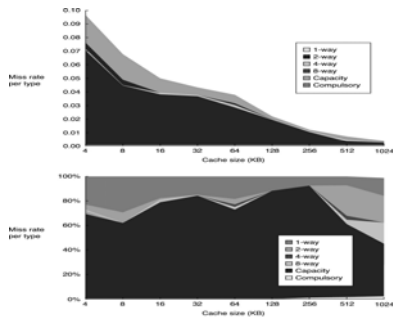
31

1. Larger Block Size

- Effects of larger block sizes
 - Reduction of compulsory misses
 - Spatial locality
 - Increase of miss penalty (transfer time)
 - Reduction of number of blocks
 - Potential increase of conflict misses
- Latency and bandwidth of lower-level memory
 - High latency and bandwidth => large block size
 - Small increase in miss penalty

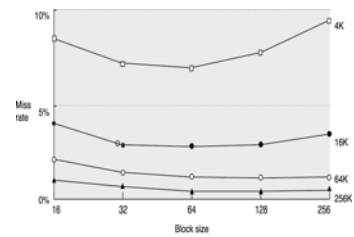
34

Miss Rates: Absolute and Distribution



32

Example



© 2003 Elsevier Science (USA). All rights reserved.

35

Reducing the Miss Rates

1. Larger block size
2. Larger Caches
3. Higher associativity
4. Pseudo-associative caches
5. Compiler optimizations

33

2. Larger Caches

- More blocks
- Higher probability of getting the data
- Longer hit time and higher cost
- Primarily used in 2nd level caches

36

3. Higher Associativity

- Eight-way set associative is good enough
- 2:1 Cache Rule:
 - Miss Rate of direct mapped cache size N = Miss Rate 2-way cache size $N/2$
- Higher Associativity can increase
 - Clock cycle time
 - Hit time for 2-way vs. 1-way external cache +10%, internal + 2%

37

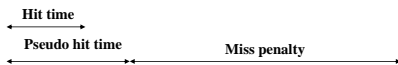
5. Compiler Optimizations

- Avoid hardware changes
- Instructions
 - Profiling to look at conflicts between groups of instructions
- Data
 - *Loop Interchange*: change nesting of loops to access data in order stored in memory
 - *Blocking*: Improve temporal locality by accessing “blocks” of data repeatedly vs. going down whole columns or rows

40

4. Pseudo-Associative Caches

- Fast hit time of *direct mapped* and lower conflict misses of 2-way *set-associative* cache?
- Divide cache: on a miss, check other half of cache to see if there, if so have a pseudo-hit (slow hit)



- Drawback:
 - CPU pipeline design is hard if hit takes 1 or 2 cycles
 - Better for caches not tied directly to processor (L2)
 - Used in MIPS R1000 L2 cache, similar in UltraSPARC

38

Loop Interchange

```

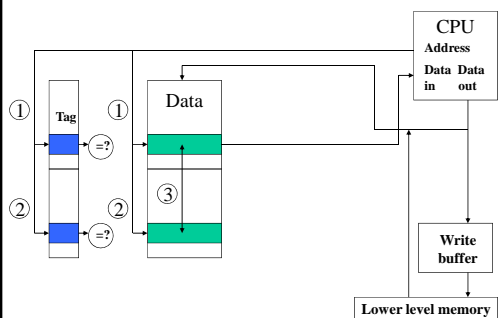
/* Before */
for (j = 0; j < 100; j = j+1)
  for (i = 0; i < 5000; i = i+1)
    x[i][j] = 2 * x[i][j];

/* After */
for (i = 0; i < 5000; i = i+1)
  for (j = 0; j < 100; j = j+1)
    x[i][j] = 2 * x[i][j];
    
```

- Sequential accesses instead of striding through memory every 100 words; improved spatial locality
- Same number of executed instructions

41

Pseudo Associative Cache



39

Blocking (1/2)

```

/* Before */
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1){
    r = 0;
    for (k = 0; k < N; k = k+1)
      r = r + y[i][k]*z[k][j];
    x[i][j] = r;
  };
    
```

- Two Inner Loops:
 - Read all $N \times N$ elements of z]
 - Read N elements of 1 row of y] repeatedly
 - Write N elements of 1 row of x]
- Capacity Misses a function of N & Cache Size:
 - $3 \times N \times 4 \Rightarrow$ no capacity misses
- Idea: compute on $B \times B$ submatrix that fits

42

Blocking (2/2)

```

/* After */
for (jj = 0; jj < N; jj = jj+B)
  for (kk = 0; kk < N; kk = kk+B)
    for (i = 0; i < N; i = i+1)
      for (j = jj; j < min(jj+B-1,N); j=j+1){
        r = 0;
        for(k=kk; k<min(kk+B-1,N);k =k+1)
          r = r + y[i][k]*z[k][j];
        x[i][j] = x[i][j] + r;
      };

```

• **B** called *Blocking Factor*

43

2. Hardware Prefetching

- Instruction Prefetching
 - Alpha 21064 fetches 2 blocks on a miss
 - Extra block placed in *stream buffer*
 - On miss check stream buffer
- Works with data blocks too:
 - 1 data stream buffer gets 25% misses from 4KB DM cache; 4 streams get 43%
 - For scientific programs: 8 streams got 50% to 70% of misses from 2 64KB, 4-way set associative caches
- Prefetching relies on having extra memory bandwidth that can be used without penalty

46

Reducing Cache Miss Penalty or Miss Rate via Parallelism

1. Nonblocking Caches
2. Hardware Prefetching
3. Compiler controlled Prefetching

44

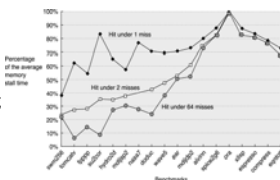
3. Compiler-Controlled Prefetching

- Compiler inserts data prefetch instructions
 - Load data into register (HP PA-RISC loads)
 - Cache Prefetch: load into cache (MIPS IV, PowerPC)
 - Special prefetching instructions cannot cause faults; a form of speculative execution
- Nonblocking cache: overlap execution with prefetch
- Issuing Prefetch Instructions takes time
 - Is cost of prefetch issues < savings in reduced misses?
 - Higher superscalar reduces difficulty of issue bandwidth

47

1. Nonblocking Cache

- Out-of-order execution
 - Proceeds with next fetches while waiting for data to come



© 2003 Elsevier Science (USA). All rights reserved.

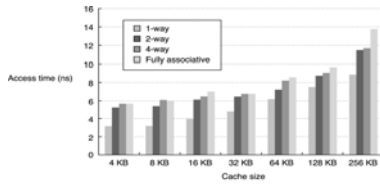
45

Reducing Hit Time

1. Small and Simple Caches
2. Avoiding address Translation during Indexing of the Cache

48

Small and Simple Caches



© 2003 Elsevier Science (USA). All rights reserved.

49

Main Memory Background

- Performance of Main Memory:
 - Latency: Cache Miss Penalty**
 - Access Time: time between request and word arrives
 - Cycle Time: time between requests
 - Bandwidth: I/O & Large Block Miss Penalty (L2)**
- Main Memory is *DRAM*: Dynamic Random Access Memory
 - Dynamic since needs to be refreshed periodically
 - Addresses divided into 2 halves (Memory as a 2D matrix):
 - RAS or Row Access Strobe
 - CAS or Column Access Strobe
- Cache uses *SRAM*: Static Random Access Memory
 - No refresh (6 transistors/bit vs. 1 transistor /bit, area is 10X)
 - Address not divided: Full address
- Size: DRAM/SRAM - 4-8
Cost & Cycle time: SRAM/DRAM - 8-16

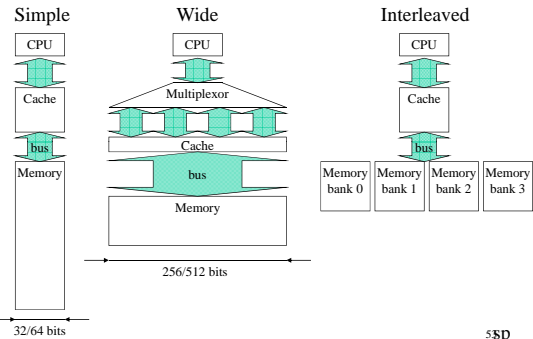
52

Avoiding Address Translation during Indexing

- Virtual vs. Physical Cache
- What happens when address translation is done
- Page offset bits to index the cache

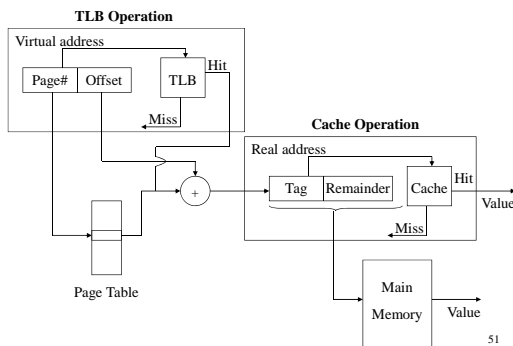
50

Main Memory Organizations



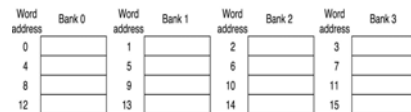
58p

TLB and Cache Operation



51

Interleaved Memory



© 2003 Elsevier Science (USA). All rights reserved.

54

Performance

- Timing model (word size is 32 bits)
 - 1 to send address,
 - 6 access time, 1 to send data
 - Cache Block is 4 words
- *Simple M.P.* = $4 \times (1+6+1) = 32$
- *Wide M.P.* = $1 + 6 + 1 = 8$
- *Interleaved M.P.* = $1 + 6 + 4 \times 1 = 11$

55