

EEC 485 High Performance Computer Architecture (Fall 2006)

Chapter 7.3 Improving Cache Performance

Chansu Yu

Cleveland State University

Cache Performance

□ Performance equation

execution time = (execution cycles + stall cycles) x cycle time

stall cycles = read-stall cycles + write-stall cycles

read-stall cycles = reads/program x read miss rate x read miss penalty

write-stall cycles = writes/program x write miss rate x write miss penalty
+ write buffer stalls

□ Two ways of improving performance:

- decreasing the miss ratio
- decreasing the miss penalty

“Decreasing the miss ratio by more flexible placement of blocks” is the main theme of this section

Direct-Mapped Caches

Advantage:

Simple → Fast & Inexpensive

Disadvantage:

Vulnerable to thrashing.

Two heavily used memory blocks map to same cache block index.

Each is repeatedly evicted to make room for the other.

Thrashing Example

```
float dot_prod(float x[SIZE],
               float y[SIZE])
{
    float sum = 0.0;
    int i;

    for (i = 0; i < SIZE; i++)
        sum += x[i]*y[i];

    return sum;
}
```

If $x[i]$ and $y[i]$ map to same blocks → thrash.

What is hit rate in this case?

Cache Performance: Tradeoffs

- (1) Increasing block size
 - + decreases miss rate, until block gets large (spatial locality)
 - increases miss penalty

- (2) Increasing cache size
 - + decreases miss rate
 - increases hit time
 - increases hardware cost

- (3) Increasing associativity (Section 7.3)
 - + increases hit rate
 - increases hit time
 - increases hardware cost

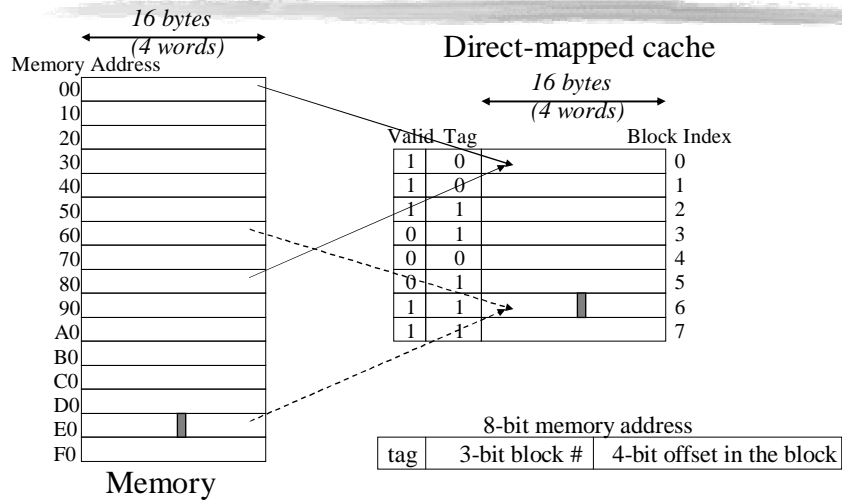
To make exact tradeoffs, need to know specific numbers.
Calculation & measurement
See book for formulae.

Other Block Placement Policies

- Direct-mapped
 - A memory block is cached in “only one” position in the cache
- Set-associative (n-way)
 - A memory block is cached in “n” positions in the cache
- Fully-associative
 - A memory block is cached in “any” position in the cache

Do they decrease miss ratio ?

Direct-mapped Block Placement

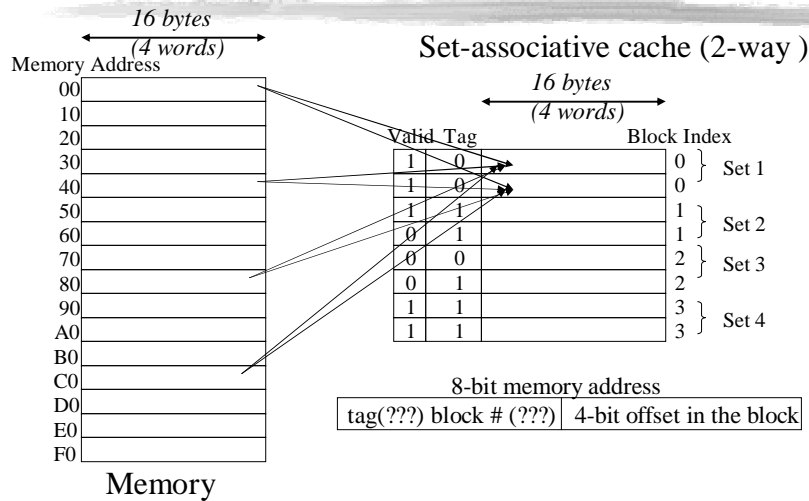


Cleveland State University

7

c.yu91@csuohio.edu

(N-Way) Set-associative Block Placement

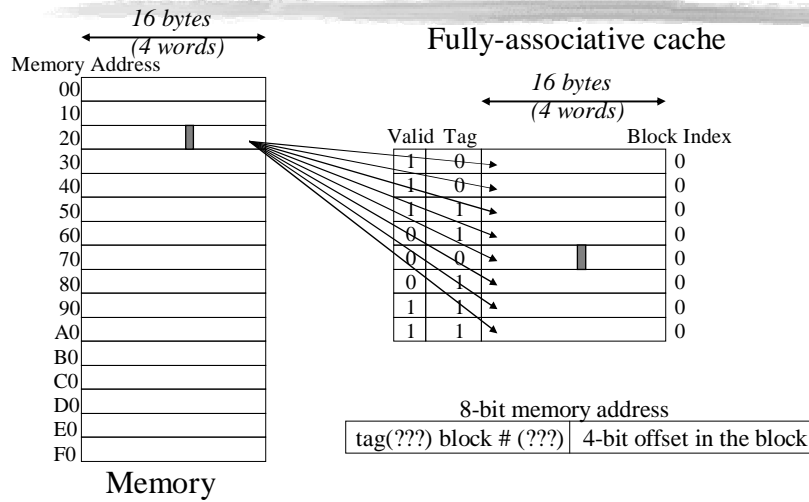


Cleveland State University

8

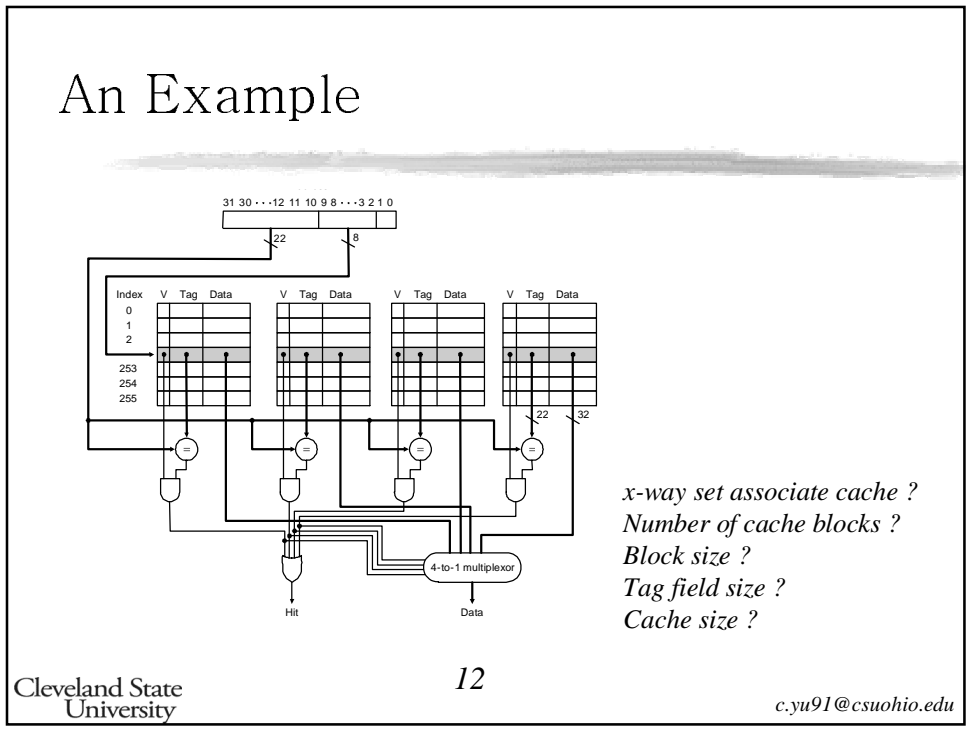
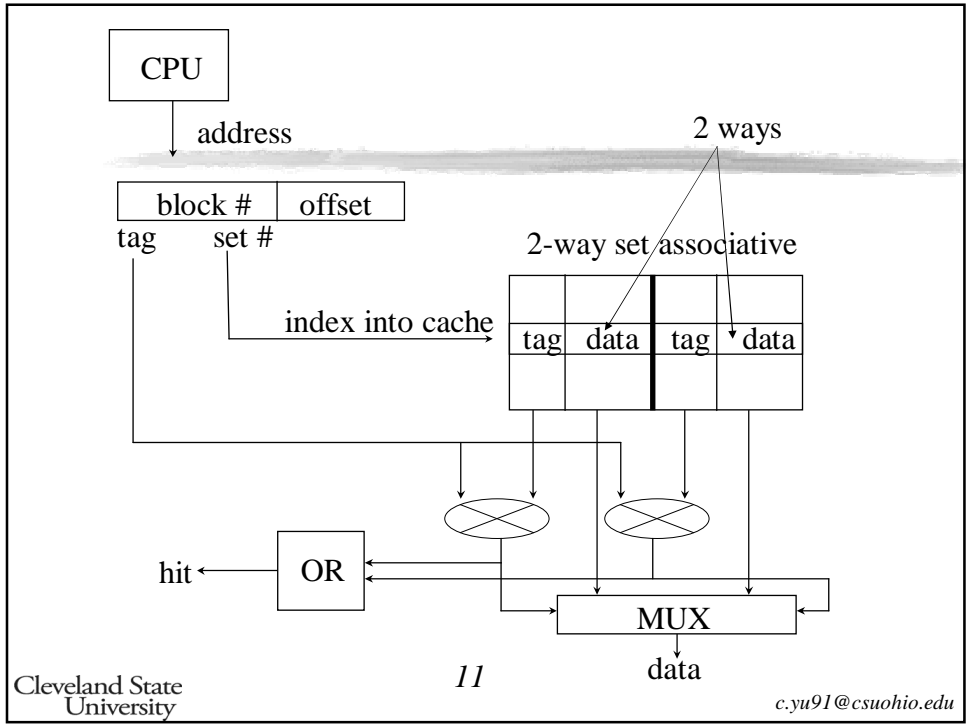
c.yu91@csuohio.edu

Fully-associative Block Placement

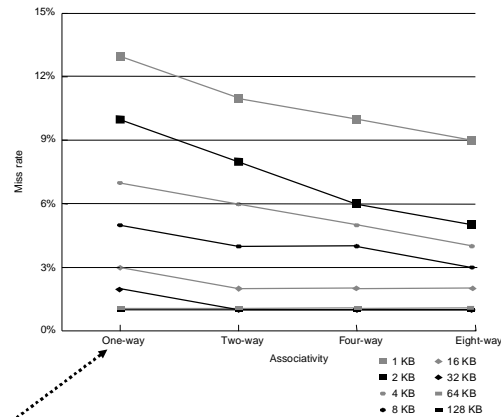


Block Identification (cont'd)

- ❑ N-way set-associative cache
 - There are N cache blocks to compare
 - N tag comparisons are done in parallel
 - “Block #” to “Set #”
 - choose low-order bits of blocks as set #
 - tag + set# + offset
 - consecutive blocks to map to different sets
 - fewer conflicts in cache, especially in the presence of spatial locality
 - Same cache size with higher associativity
 - #blocks / set ???
 - index size , tag size ???



Cache Performance: Tradeoffs



What is it ?

Replacement Policies

- ❑ In direct-mapped cache, no replacement policy is necessary
- ❑ In set-associative cache, an important question is
 - Which block to replace among the set (see page 504)?
- ❑ Least recently used (LRU)
 - The most commonly used scheme
 - How to keep track of usage of blocks?
 - Single bit in case of two-way set associative cache
 - See Section 7.5 for higher associativity case