Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Unit 4: Cache Optimization

Dr. D. C. Kiran

Department of Computer Science and Engineering

Syllabus

~~**Unit 1: Basic Processor Architecture and Design**~~

~~**Unit 2: Pipelined Processor and Design**~~

~~**Unit 3: Memory**~~

Unit 4: Input/Output Device Design

- ~~3 C~~
- ~~Introduction to Cache Optimization~~
- **Recuse Miss Rate**

Unit 5: Advanced Architecture

Reducing Miss Rate!!!

Optimization 1: Large Block Size

Optimization 2: Large Cache Capacity

Optimization 3: Higher Associativity

Optimization 1: Large Block Size to Reduce Miss Rate

Advantage:

- Satisfy Spatial Locality
- Reduces Compulsory Misses

Example: Instead of 4 word Block, use 8 word Block or 16 word Block

Optimization 1: Large Block Size to Reduce Miss Rate

Disadvantage:

- Increase Miss Penalty (Bus width Issues)

Example: If Width of Bus is 4 word, 2 transfers for 8 word Block and 4 transfers for 16 word Block are required.

- Increase Conflict Miss, as Number of Cache Blocks reduces due increase in Block Size.

Example: A 32 word cache can be designed as

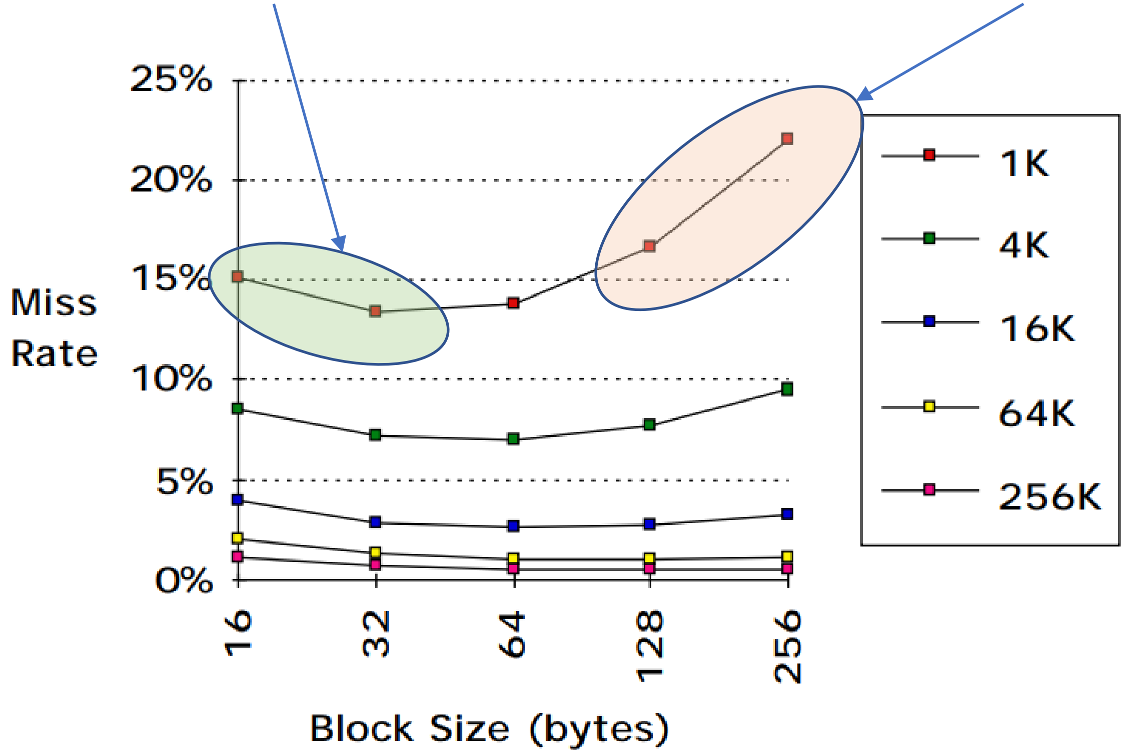
  - Eight, 4 word block
  - Four, 8 word block or
  - Two, 16 word block

- May bring useless data into cache as Spatial Locality cannot be maintained in entire program consistently.

Optimization 1: Large Block Size to Reduce Miss Rate

Reduced Compulsory Miss

Increased Conflict Miss



- Performance keeps improving to a limit.
- With fewer number of cache block, increases conflict misses and thus overall cache miss rate

Optimization 2: Large Cache to Reduce Miss Rate

Advantage:

- Reduces Capacity Miss
- Can accommodate large memory footprint.

Example: If number of Blocks are 4096:

- On a cache with 128 Lines, 32 possible Blocks are mapped under direct mapping.

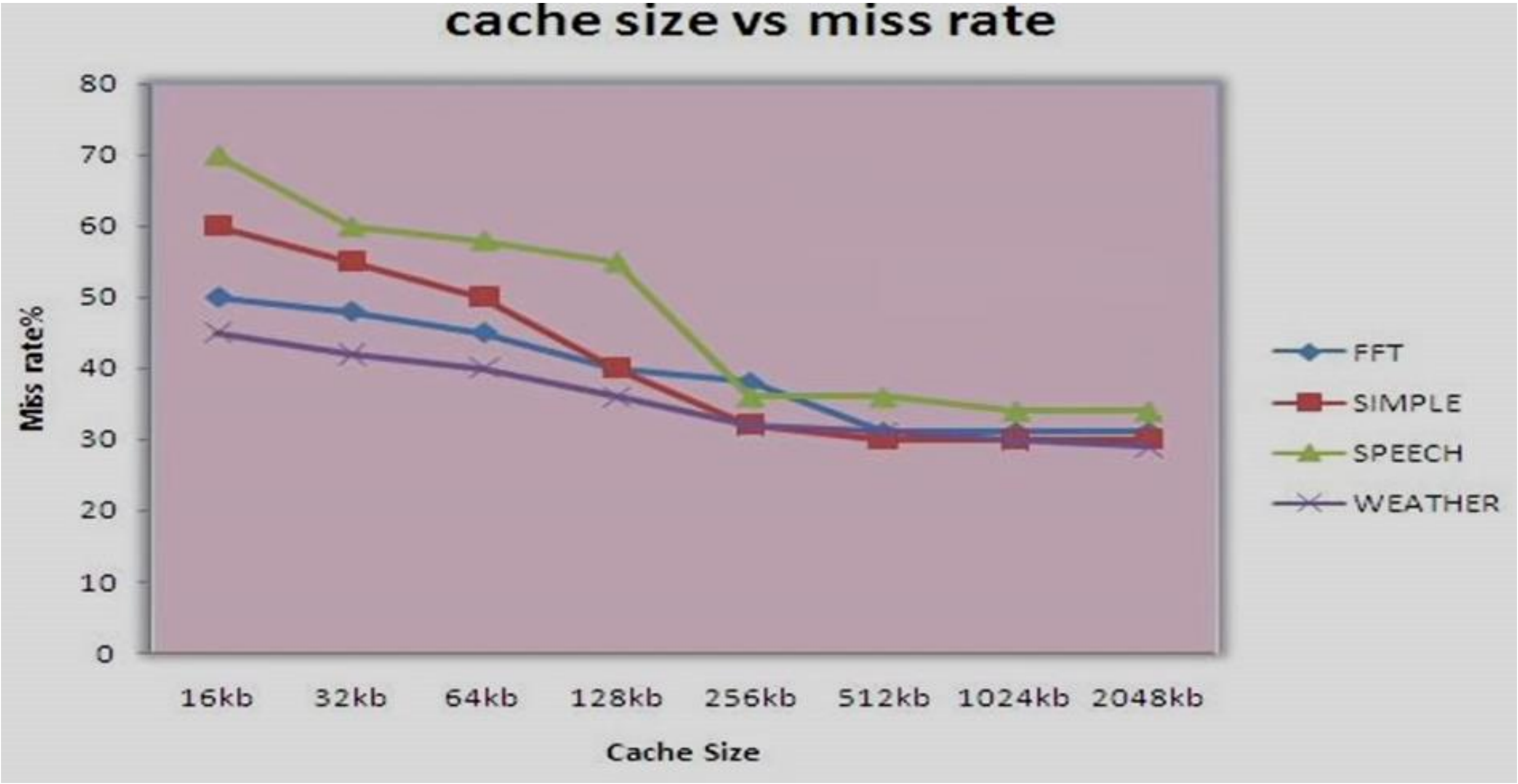
- On a cache with 256 Lines, Only 16 possible Blocks are mapped under direct mapping.

Optimization 2: Large Cache to Reduce Miss Rate

Disadvantage:

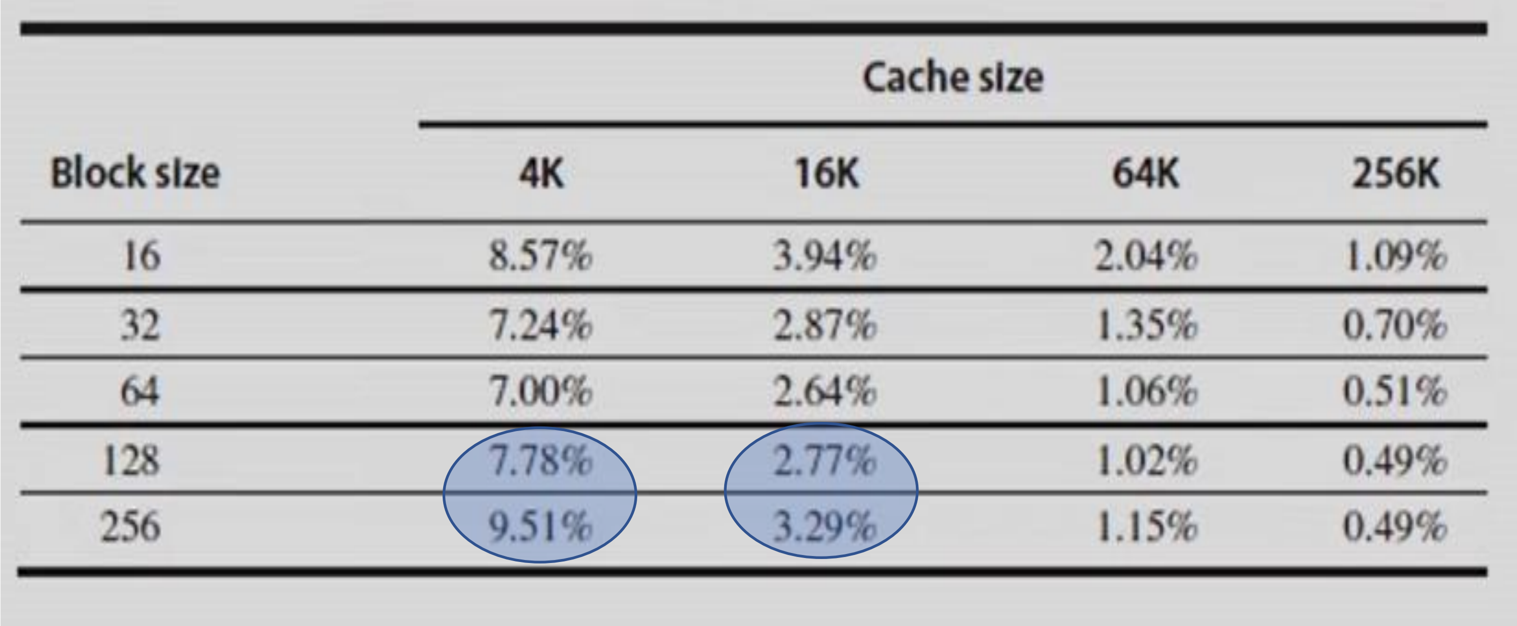
- Increases Hit Time
- High Cost, Area and Power

Optimization 2: Large Cache to Reduce Miss Rate



Optimization 1 & 2: Reduce Miss Rate

Block Size vs Cache Size



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

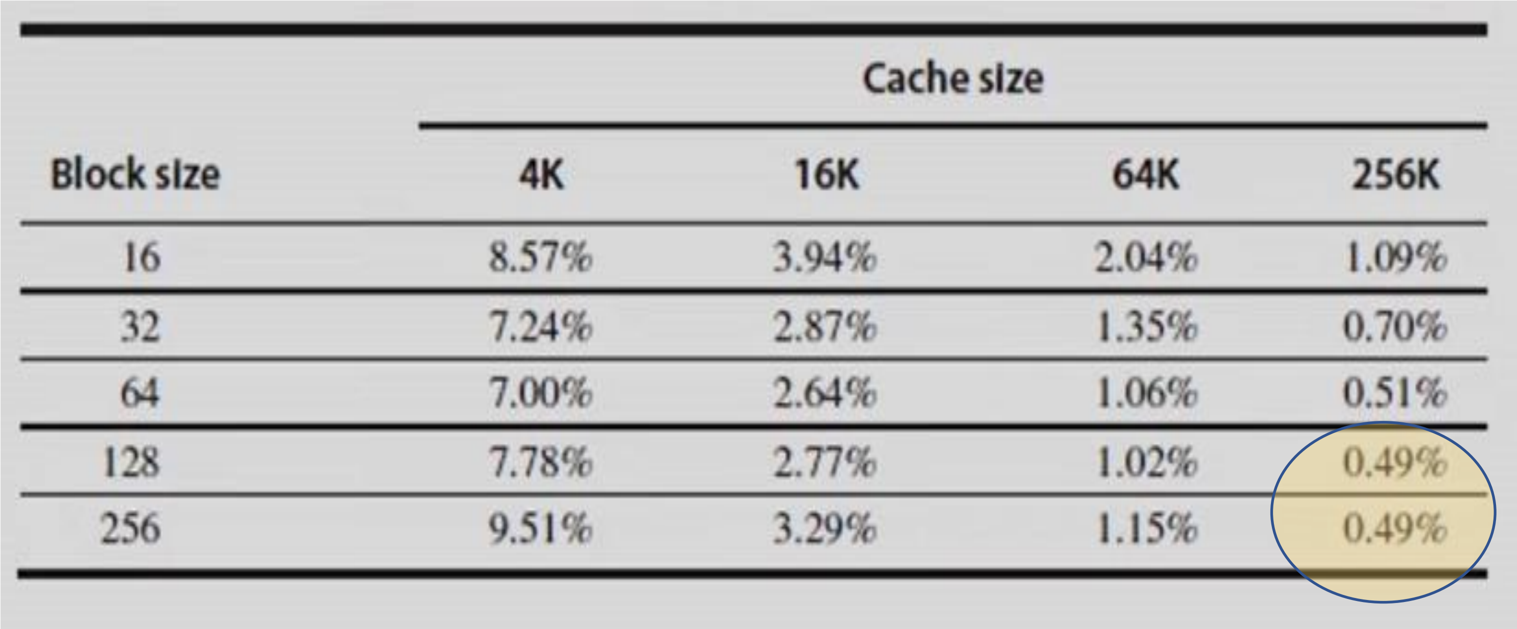
Observation1:

Miss Rate Increases if Cache Size is small and Block size is Large

4 K Cache with 256 Block size has highest Miss Rate

Optimization 1 & 2: Reduce Miss Rate

Block Size vs Cache Size



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

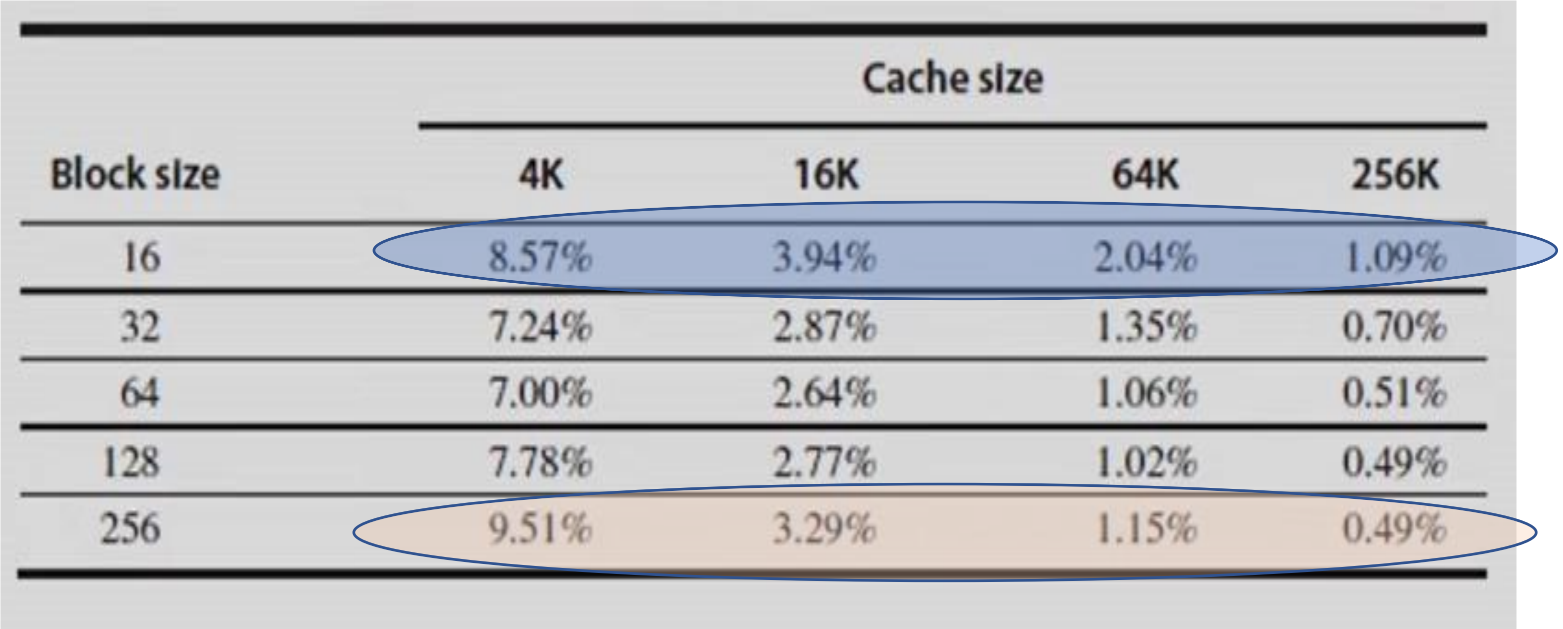
Observation2:

Miss Rate Decreases if Cache Size is Large and Block size is Large

256 k Cache with 256 Block Size has less Miss Rate

Optimization 1 & 2: Reduce Miss Rate

Block Size vs Cache Size



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Observation 3:

Miss Rate Decreases if Cache Size is Large and Block size is Small

Miss Rate Decreases if Cache Size is Large and Block Size is Large

However, If Cache Size if large, Hit Time, Space, Power and Cost Increases.

Optimization 1 & 2: Reduce Miss Rate

Example 1: Assume the memory system takes 80 clock cycles of overhead and then delivers 16 bytes every 2 clock cycles. That is, it can supply 16 bytes in 82 clock cycles, 32 bytes in 84 clock cycles, and so on… Which block size has the smallest average memory access time for each cache size?

Ans: Average access time = Hit time + Miss rate x Miss penalty

If we assume that the hit time is 1 clock cycle independent of the block size, then,

The access time for a 16- byte block in a **4KB cache is**

Average access time = 1 + (8.57% x 82)

= 1 + 7.0274

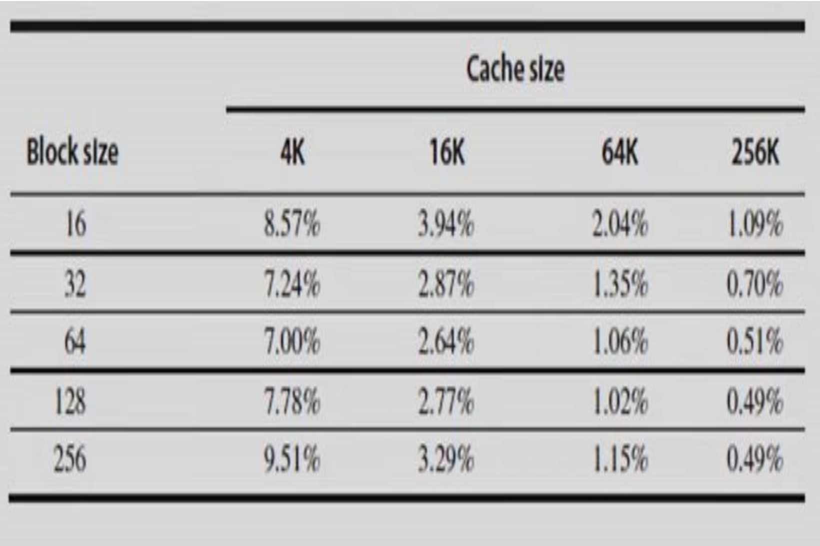
= **8.0274 clock cycles.**

For, 256 byte block in a 256 KB cache the

Average access time = 1 + (0.49% x (80+32))

= 1 + (0.5488)

= **1.5488 clock cycles.**



	Cache size			
Block size	4K	16K	64K	256K
16	8.57%	3.94%	2.04%	1.09%
32	7.24%	2.87%	1.35%	0.70%
64	7.00%	2.64%	1.06%	0.51%
128	7.78%	2.77%	1.02%	0.49%
256	9.51%	3.29%	1.15%	0.49%

Optimization 3: Large Associativity to Reduce Miss Rate

Higher Associativity to Reduce Miss Rate

- Fully Associative cache are best, as no replacement is required till cache is full.
- Set Associative cache balances the Search Time and Replacement.

Advantage:

- Reduce Conflict Miss
- Reduce Miss Rate and Eviction rate.

Disadvantage:

- Increases Hit Time (Due to increase in search time by comparing each TAG)
- Complex design than Direct Map.

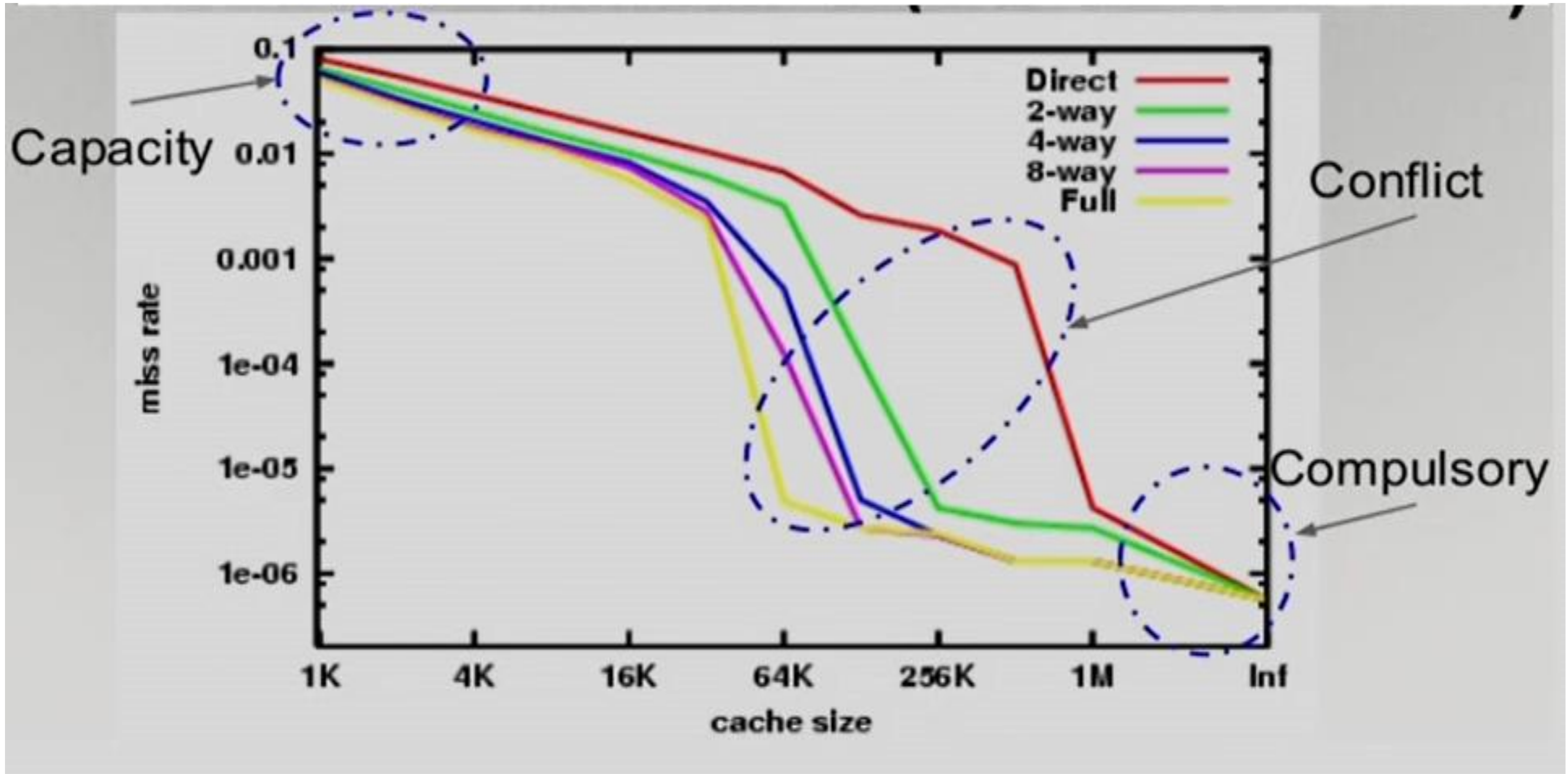
Replacement

- 2-way is better than Direct Mapping
- 4-way is better than 2-way
- 8-way is better than 4-way,

Search time

8-way, 16-way may become closer to Fully Associative.

Optimization 3: Large Associativity to Reduce Miss Rate



Replacement

- 2-way is better than Direct Mapping
- 4-way is better than 2-way
- 8-way is better than 4-way,

Search time

8-way, 16-way may become closer to Fully Associative.

Optimization 3: Large Associativity to Reduce Miss Rate

What about AMAT?

AMAT vs Cache Associativity

Assumption 1: Higher Associativity would increase the clock cycle time and is as listed below:

$\text{Clock Cycle Time}_{2\_ \text{way}} = 1.36 \times \text{Clock Cycle Time}_{1\_\text{way}}$

$\text{Clock Cycle Time}_{4\_ \text{way}} = 1.42 \times \text{Clock Cycle Time}_{1\_\text{way}}$

$\text{Clock Cycle Time}_{8\_ \text{way}} = 1.52 \times \text{Clock Cycle Time}_{1\_\text{way}}$

Assumption 2: Hit time is 1 Clock and Miss Penalty= 25 Clock Cycle

AMAT= Hit Time + Miss Rate x Miss Penalty

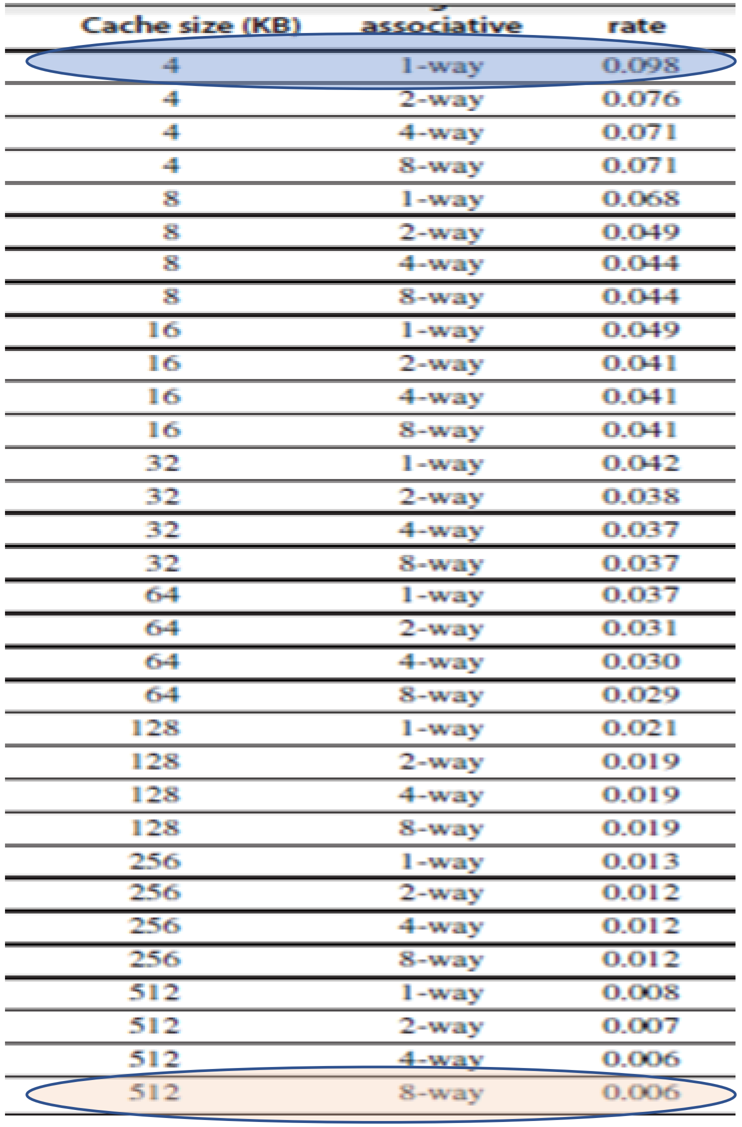
$\text{access time}_{8\text{-way}} = 1.52 + \text{Miss Rate} \times 25$

$\text{access time } 4_{\text{-way}} = 1.44 + \text{Miss Rate} \times 25$

$\text{access time } 2_{\text{-way}} = 1.36 + \text{Miss Rate} \times 25$

$\text{access time}_{1\text{way}} = 1.00 + \text{Miss Rate} \times 25$

Optimization 3: Large Associativity to Reduce Miss Rate



Cache size (KB)	associative	rate
4	1-way	0.098
4	2-way	0.076
4	4-way	0.071
4	8-way	0.071
8	1-way	0.068
8	2-way	0.049
8	4-way	0.044
8	8-way	0.044
16	1-way	0.049
16	2-way	0.041
16	4-way	0.041
16	8-way	0.041
32	1-way	0.042
32	2-way	0.038
32	4-way	0.037
32	8-way	0.037
64	1-way	0.037
64	2-way	0.031
64	4-way	0.030
64	8-way	0.029
128	1-way	0.021
128	2-way	0.019
128	4-way	0.019
128	8-way	0.019
256	1-way	0.013
256	2-way	0.012
256	4-way	0.012
256	8-way	0.012
512	1-way	0.008
512	2-way	0.007
512	4-way	0.006
512	8-way	0.006

AMAT vs Cache Associativity

Example

The time for a 4KB Direct–Mapped cache is

Average memory access time $_{1\text{-way}}$ = 1.00 + (0.098 x 25) = 3.45

The time for a 512 KB, Eight-Way Set Associative cache is

Average memory access time $_{8\text{-way}}$ = 1.52 + (0.006 x 25) = 1.67

Example will make us to believe that

Average memory access time$_{8\text{-way}}$ < Average memory access time$_{4\text{-way}}$

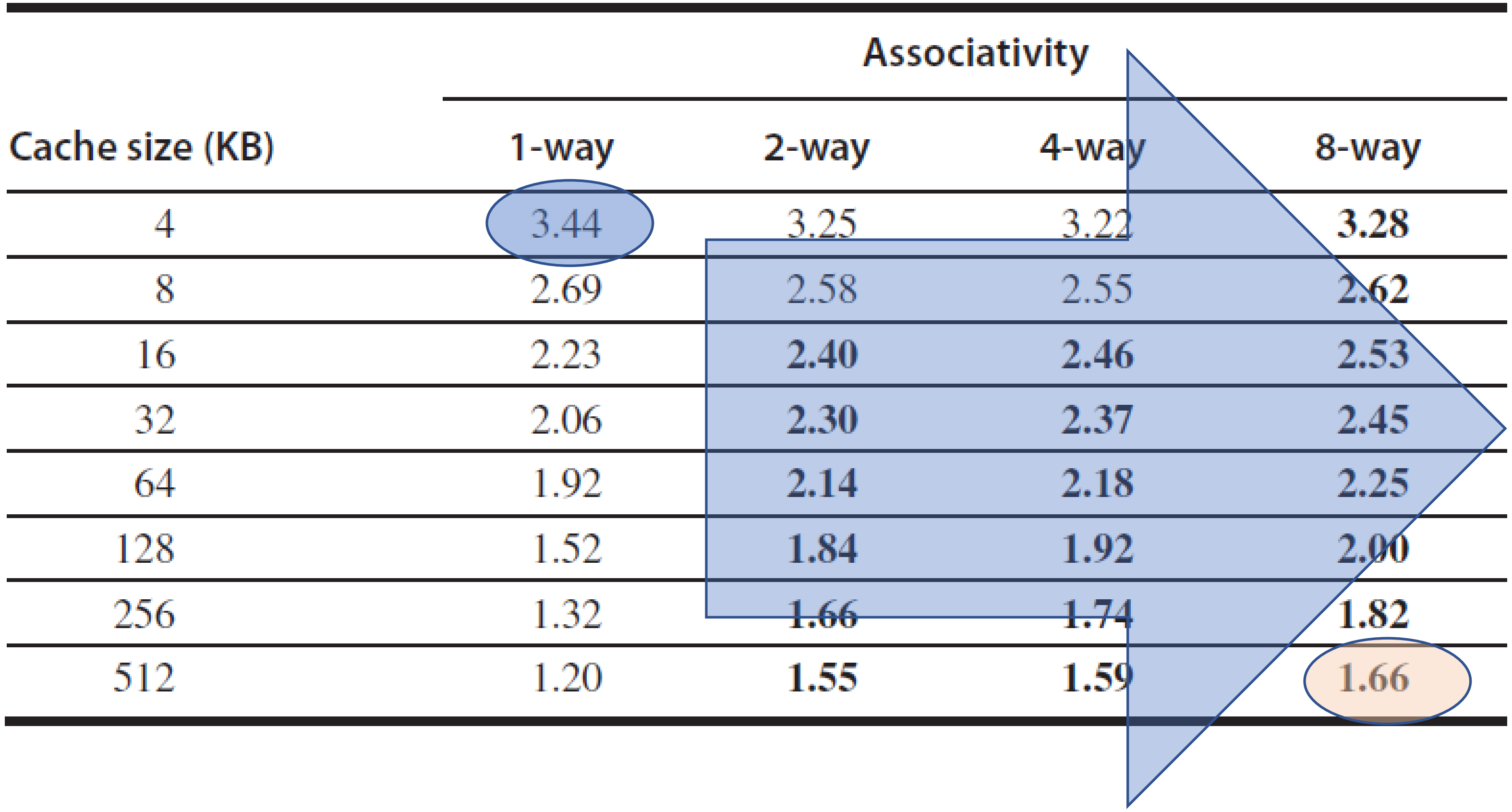
Average memory access time$_{4\text{-way}}$ < Average memory access time$_{2\text{-way}}$

Average memory access time$_{2\text{-way}}$ < Average memory access time$_{1\text{-way}}$

However ☹

Optimization 3: Large Associativity to Reduce Miss Rate

AMAT vs Cache Associativity



	Associativity			
Cache size (KB)	1-way	2-way	4-way	8-way
4	3.44	3.25	3.22	**3.28**
8	2.69	2.58	2.55	**2.62**
16	2.23	**2.40**	**2.46**	**2.53**
32	2.06	**2.30**	**2.37**	**2.45**
64	1.92	**2.14**	**2.18**	**2.25**
128	1.52	**1.84**	**1.92**	**2.00**
256	1.32	**1.66**	**1.74**	**1.82**
512	1.20	**1.55**	**1.59**	1.66

High Associativity Leads to Higher Access Time

Next Session

Reducing Miss Penalty

THANK YOU

Dr. D. C. Kiran

Department of Computer Science and Engineering

dckiran@pes.edu

9829935135