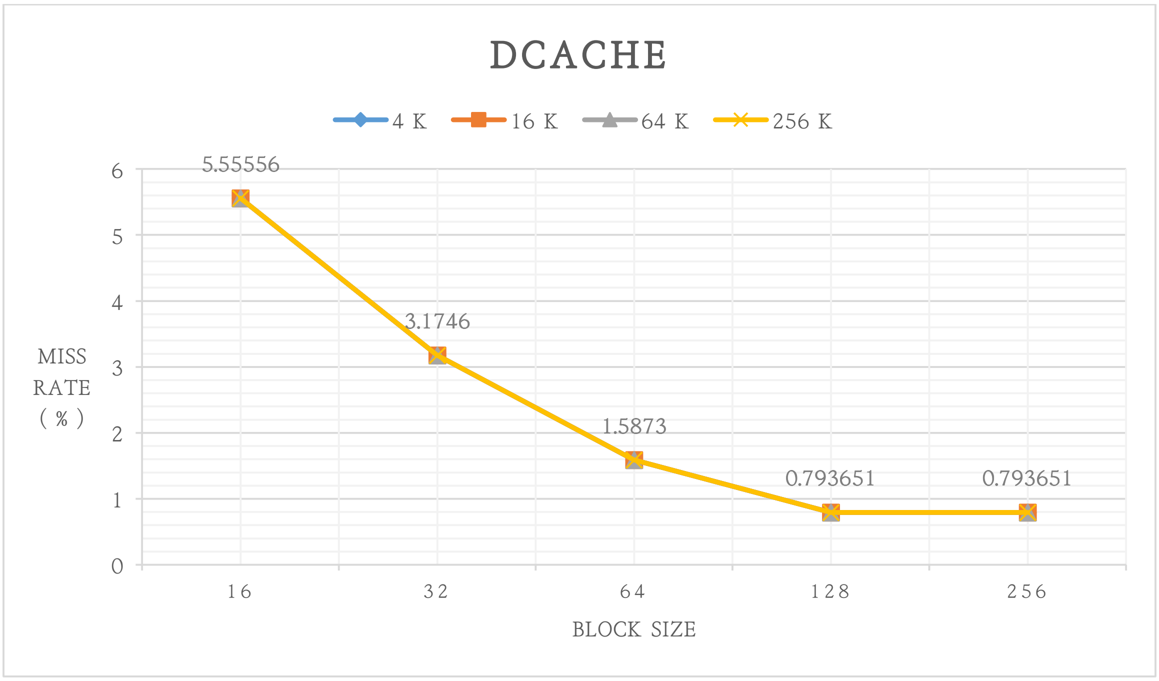
Computer Organization

Lab 4 Report : Cache Simulator

0513311 羅文慧

- [Basic Problem]

1. DCACHE :



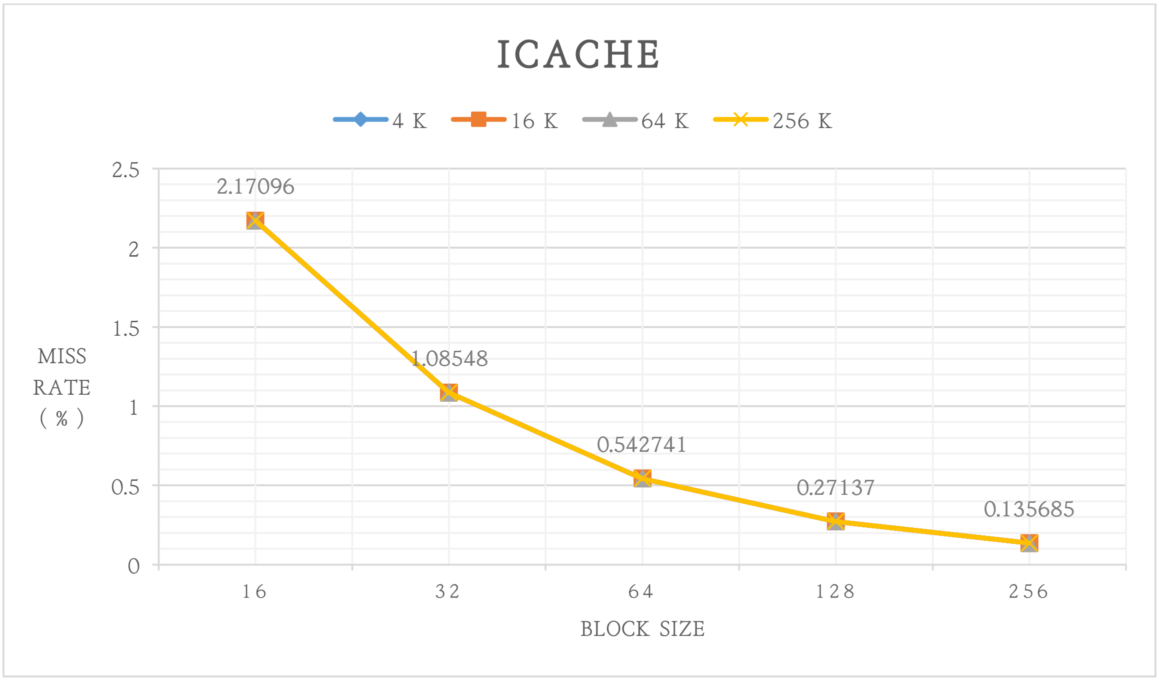
Block size / Cache size	16	32	64	128	256
4 K	5.55556 %	3.1746 %	1.5873 %	0.793651 %	0.793651 %
16 K	5.55556 %	3.1746 %	1.5873 %	0.793651 %	0.793651 %
64 K	5.55556 %	3.1746 %	1.5873 %	0.793651 %	0.793651 %
256 K	5.55556 %	3.1746 %	1.5873 %	0.793651 %	0.793651 %

Once the cache size is given, Larger Block Size -> Raise Spatial Locality -> Lower Miss Rate.

But if we raise the block size unlimitedly, number of the total blocks in cache would be fewer

-> need to transfer more data -> raise the miss penalty.

2. ICACHE :



Block size / Cache size	16	32	64	128	256
4 K	2.17096 %	1.08548 %	0.542741 %	0.27137 %	0.135685 %
16 K	2.17096 %	1.08548 %	0.542741 %	0.27137 %	0.135685 %
64 K	2.17096 %	1.08548 %	0.542741 %	0.27137 %	0.135685 %
256 K	2.17096 %	1.08548 %	0.542741 %	0.27137 %	0.135685 %

Once the cache size is given, Larger Block Size -> Raise Spatial Locality -> Lower Miss Rate.

But if we raise the block size unlimitedly, number of the total blocks in cache would be fewer

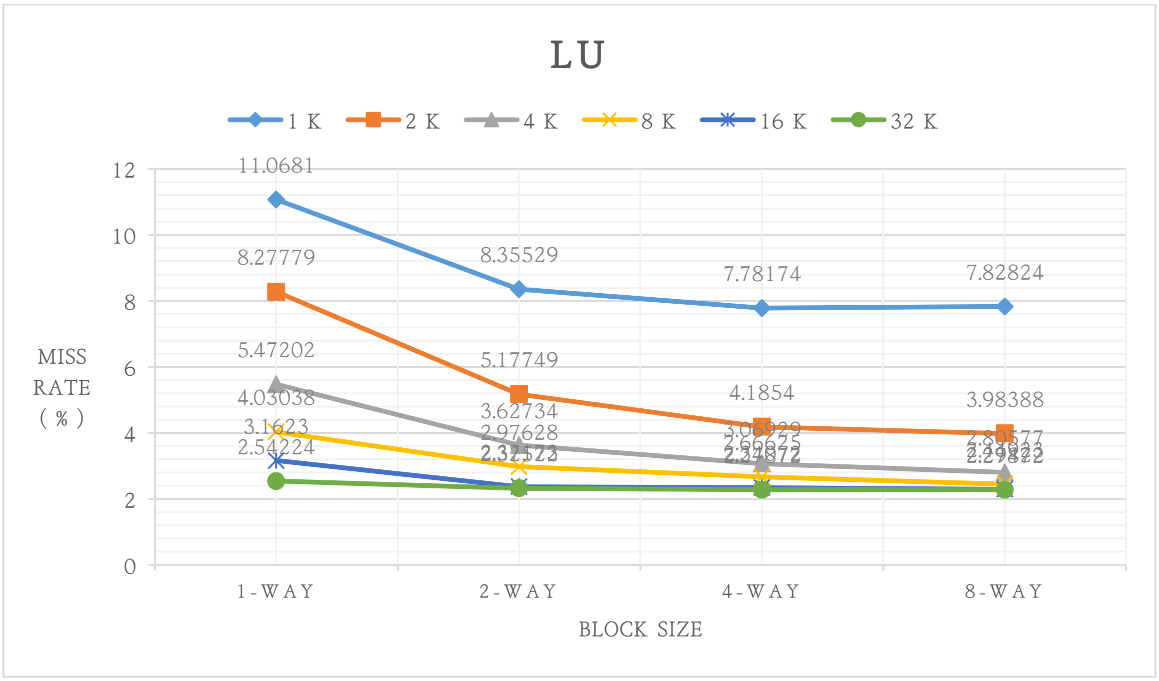
-> need to transfer more data -> raise the miss penalty.

Miss rate in I-Cache is usually lower than D-Cache, because of the behavior of different caches :

I-Cache is almost visited in sequence while D-Cache is visited in various patterns.

- [Advanced Problem]

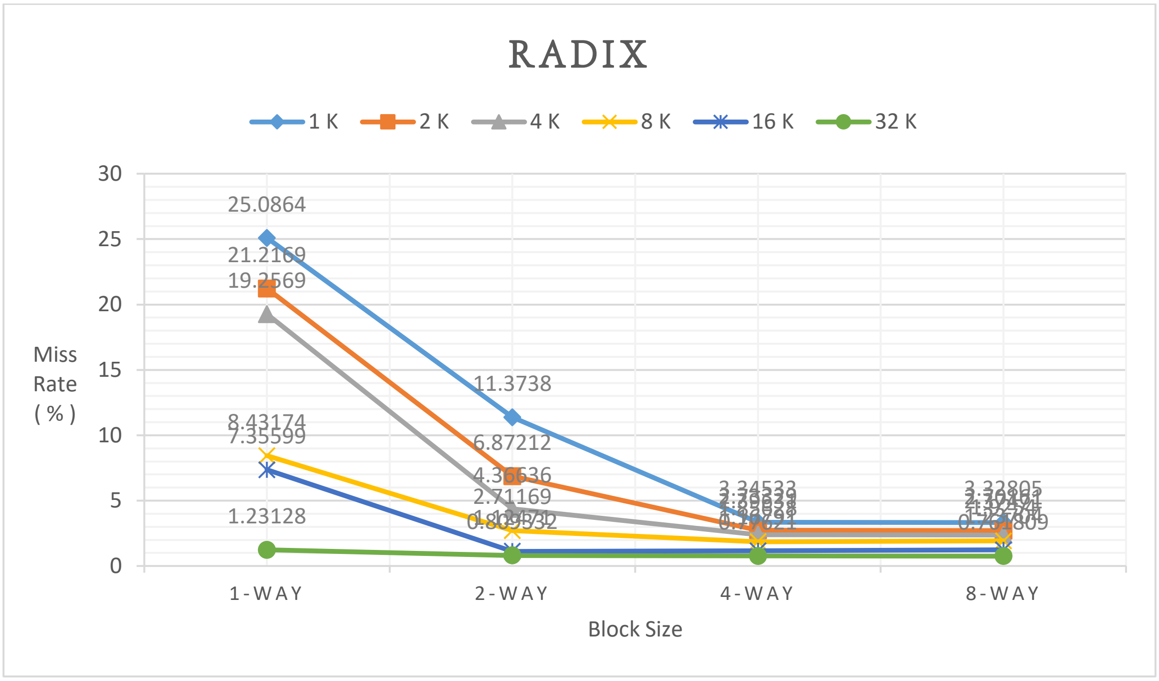
3. LU :



Associativity / Cache size	1-way	2-way	4-way	8-way
1 K	11.0681 %	8.35529 %	7.78174 %	7.82824 %
2 K	8.27779 %	5.17749 %	4.1854 %	3.98388 %
4 K	5.47202 %	3.62734 %	3.06929 %	2.80577 %
8 K	4.03038 %	2.97628 %	2.66625 %	2.44923 %
16 K	3.1623 %	2.37173 %	2.34072 %	2.29422 %
32 K	2.54224 %	2.32522 %	2.27872 %	2.27872 %

Lower cache size or associativity -> increase miss rate.
But if we raise the block size unlimitedly, number of the total blocks in cache would be fewer -> need to transfer more data -> raise the miss penalty.
(i.e. cache size 1K)

4. RADIX :



Associativity / Cache size	1-way	2-way	4-way	8-way
1 K	25.0864 %	11.3738 %	3.34533 %	3.32805 %
2 K	21.2169 %	6.87212 %	2.73329 %	2.70161 %
4 K	19.2569 %	4.36636 %	2.39631 %	2.37471 %
8 K	8.43174 %	2.71169 %	1.85628 %	1.9254 %
16 K	7.35599 %	1.12471 %	1.16791 %	1.23704 %
32 K	1.23128 %	0.809332 %	0.77621 %	0.761809 %

Lower cache size or associativity -> increase miss rate.
But if we raise the block size unlimitedly, number of the total blocks in cache would be fewer -> need to transfer more data -> raise the miss penalty.
(i.e. cache size 8K & 16K)

Associativity / Total bits	1-way	2-way	4-way	8-way
1 K	8,560	8,576	8,592	8,608
2 K	17,088	17,120	17,152	17,184
4 K	34,112	34,176	34,240	34,304
8 K	68,096	68,224	68,352	68,480
16 K	135,936	136,192	136,448	136,704
32 K	271,360	271,872	272,384	272,896

• Detailed description of the implementation :

1. CPU compared to Lab3 :

   (1) Enlarge the Instruction Memory to fit the input size of test data.

   (2) Modify the wire and module name in CPU.v to fit the given testbench.

2. Cache Simulator :

   (1) direct_mapped_cache.cpp :

   Record total number of hit and miss case, and output the miss rate = miss / (miss +hit) in percentage.

   (2) direct_mapped_cache_lru.cpp :

   Use struct to build the Data which consist of valid bit, tag, and time; because we need to discuss N-way associativity, so I use another struct to build the data[N] in Data type.

   Use LRU means we have to replace the least-recently used data with the new one, so I add a time-stamp. Whenever there is a new data needed to be added in cache, run through the whole cache to find the least number in time-stamp, replace it with the new one.