Computer Organization Lab4

0616015 劉姿利、0616092 粘捷

How to Compile

Just type `make`, which will execute the `Makefile` below.

parameters:

```
V=iverilog
VTARGET=code/verilog/*.v
VOUT=code/verilog/CPU.vvp
TXTOUT=test/ICACHE.txt test/DCACHE.txt

CC=g++ -O2
CMAP=code/cpp/direct_mapped_cache
CLRU=code/cpp/direct_mapped_cache_lru
CBIT=code/cpp/direct_mapped_cache_lru_totalbits

OUTFILE=test/out.txt

.PHONY: clean
```

trace:

- compiles `.v` files (output: `.vvp`)
- executes `.vvp` (which will generate `I/DCACHE.txt`)

```
trace:
	${V} -o ${VOUT} ${VTARGET}
	vvp ${VOUT}
```

map and lru:

- compile `.cpp` files (output: `.out`)

```
map: trace
	${CC} -o ${CMAP}.out ${CMAP}.cpp

lru:
	${CC} -o ${CLRU}.out ${CLRU}.cpp
	${CC} -o ${CBIT}.out ${CBIT}.cpp
```

display:

- executes `.out` files
- stores the results into `out.txt`
- print out out.txt

```
display: map lru
	./${CMAP}.out > ${OUTFILE}
	./${CLRU}.out >> ${OUTFILE}
	./${CBIT}.out >> ${OUTFILE}
	cat ${OUTFILE}
```

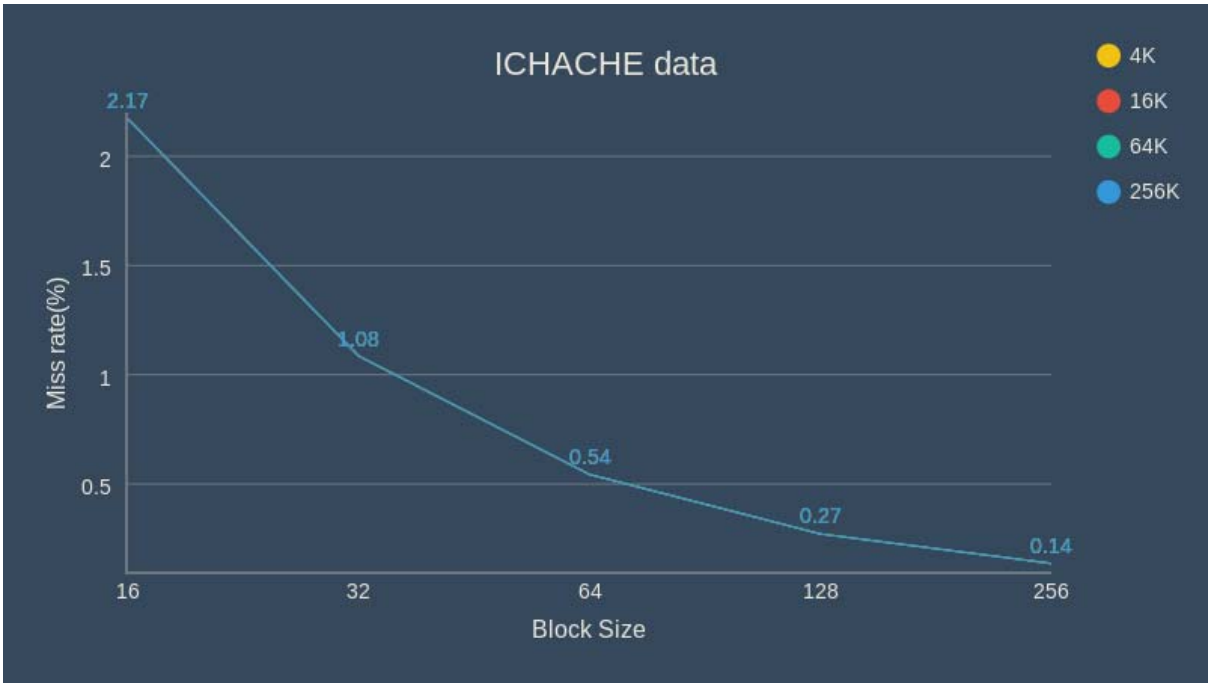
The Results

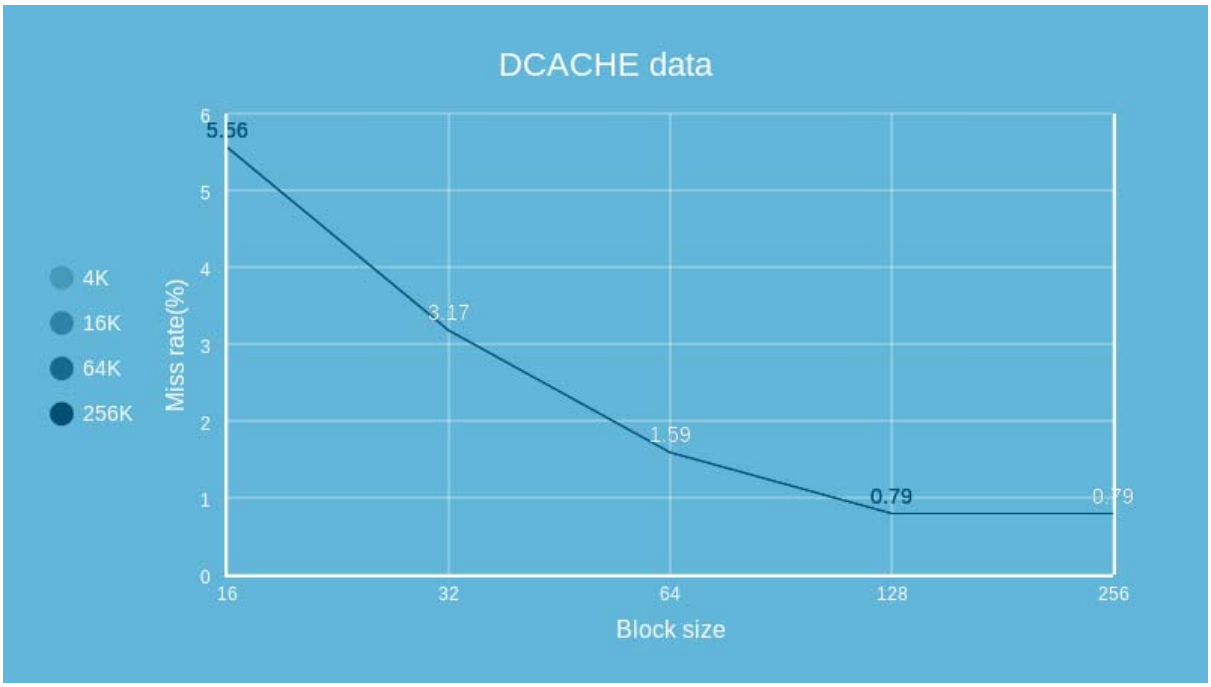
Basic

- in `out.txt`

```
=== direct_mapped_cache.cpp ===

test/ICACHE.txt

             16       32       64      128      256
--------------------------------------------------
  4K:     2.171    1.085    0.543    0.271    0.136
 16K:     2.171    1.085    0.543    0.271    0.136
 64K:     2.171    1.085    0.543    0.271    0.136
256K:     2.171    1.085    0.543    0.271    0.136

test/DCACHE.txt

             16       32       64      128      256
--------------------------------------------------
  4K:     5.556    3.175    1.587    0.794    0.794
 16K:     5.556    3.175    1.587    0.794    0.794
 64K:     5.556    3.175    1.587    0.794    0.794
256K:     5.556    3.175    1.587    0.794    0.794
```

- graph


ICHACHE data
4K
16K
64K
256K
2.17
1.08
0.54
0.27
0.14
2
1.5
1
0.5
Miss rate(%)
16
32
64
128
256
Block Size

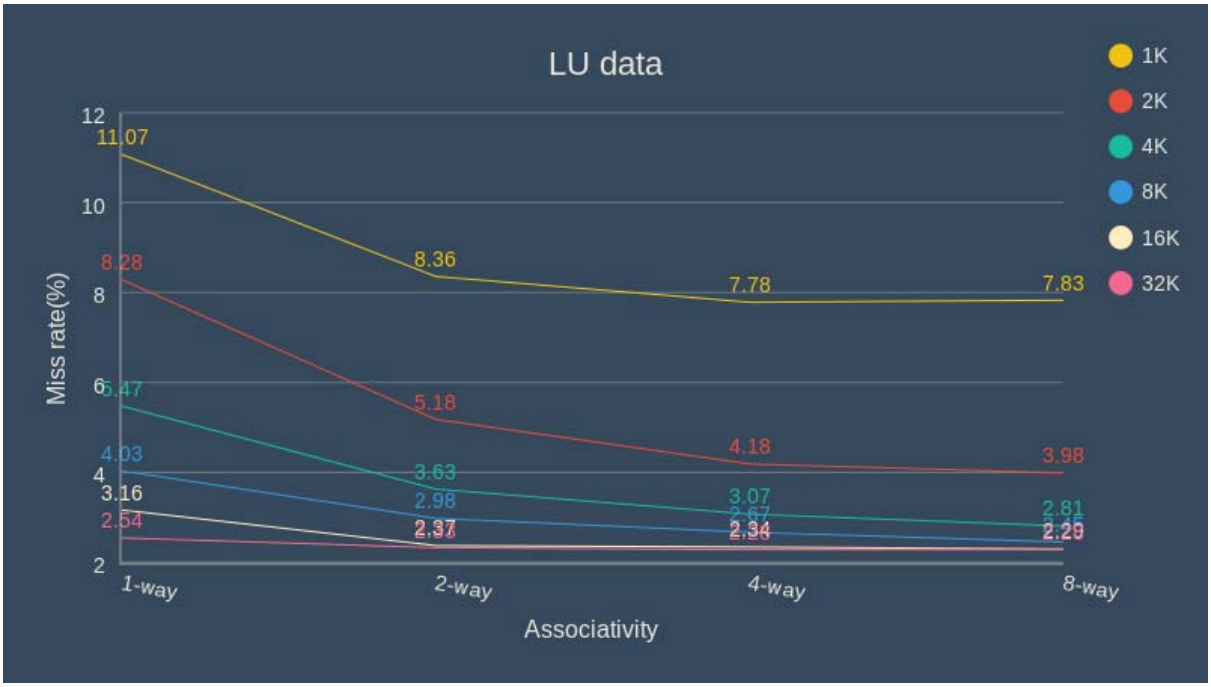

DCACHE data
4K
16K
64K
256K
5.56
3.17
1.59
0.79
0.79
6
5
4
3
2
1
0
Miss rate(%)
16
32
64
128
256
Block size

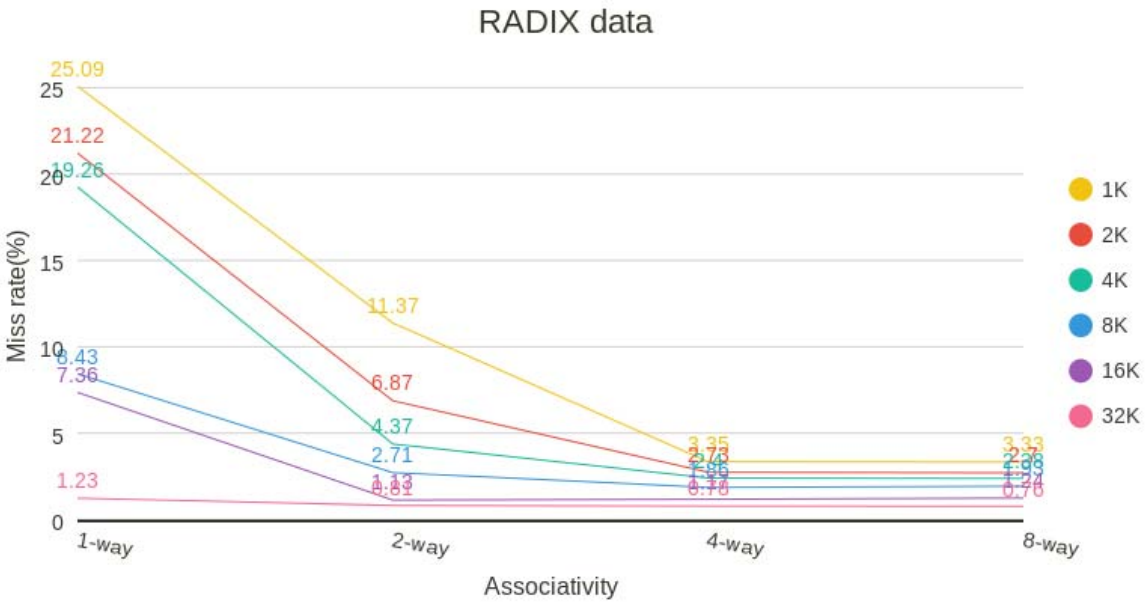
Advanced

- miss rate
  - in `out.txt`

```
=== direct_mapped_cache_lru.cpp ===

test/LU.txt

         1-way    2-way    4-way    8-way
----------------------------------------
 1K:    11.068    8.355    7.782    7.828
 2K:     8.278    5.177    4.185    3.984
 4K:     5.472    3.627    3.069    2.806
 8K:     4.030    2.976    2.666    2.449
16K:     3.162    2.372    2.341    2.294
32K:     2.542    2.325    2.279    2.279

test/RADIX.txt

         1-way    2-way    4-way    8-way
----------------------------------------
 1K:    25.086   11.374    3.345    3.328
 2K:    21.217    6.872    2.733    2.702
 4K:    19.257    4.366    2.396    2.375
 8K:     8.432    2.712    1.856    1.925
16K:     7.356    1.125    1.168    1.237
32K:     1.231    0.809    0.776    0.762
```

- graph


LU data
Miss rate(%)
12
10
8
6
4
2
11.07
8.36
7.78
7.83
8.28
5.18
4.18
3.98
6.47
3.63
2.98
3.07
2.81
4.03
3.16
2.54
1-way
2-way
4-way
8-way
Associativity
1K
2K
4K
8K
16K
32K


RADIX data
Miss rate(%)
25
20
15
10
5
0
25.09
21.22
19.26
11.37
6.87
4.37
2.71
1.23
1-way
2-way
4-way
8-way
Associativity
1K
2K
4K
8K
16K
32K

-

- compute total bits (in `out.txt`)

```
=== direct_mapped_cache_lru_totalbits.cpp ===

test/LU.txt

         1-way    2-way    4-way    8-way
----------------------------------------
 1K:      8560     8576     8592     8608
 2K:     17088    17120    17152    17184
 4K:     34112    34176    34240    34304
 8K:     68096    68224    68352    68480
16K:    135936   136192   136448   136704
32K:    271360   271872   272384   272896

test/RADIX.txt

         1-way    2-way    4-way    8-way
----------------------------------------
 1K:      8560     8576     8592     8608
 2K:     17088    17120    17152    17184
 4K:     34112    34176    34240    34304
 8K:     68096    68224    68352    68480
16K:    135936   136192   136448   136704
32K:    271360   271872   272384   272896
```

totalbits = (1 + (32 - index_bit - offset_bit) + block_size * 8) * (num_of_blocks * associativity) (bits)

- note that the unit of the given block_size is byte