캐시 결과

▼ "write-through", "write-allocate", "fifo” 일때 s,b,size 변경 효율

8_8_4 : 3.69
8_8_8 : 7.46
8_8_16 : 12.88

8_16_4 : 7.41
8_16_8 : 12.59
8_16_16 : 17.51

8_32_4 : 11.97
8_32_8 : 16.97
8_32_16 : 19.91

16_8_4 : 7.41
16_8_8 : 12.59
16_8_16 : 17.51

16_16_4 : 11.97
16_16_8 : 16.97
16_16_16 : 19.91

16_32_4 : 14.44
16_32_8 : 19.01
16_32_16 : 21.23

32_8_4 : 11.97
32_8_8 : 16.97
32_8_16 : 19.91

32_16_4 : 14.44

32_16_8 : 19.01
32_16_16 : 21.23

32_32_4 : 15.99
32_32_8 : 20.50
32_32_16 : 21.92

▼ lru, fifo, random 일때

1. lru
    - 22.26
2. fifo
    - 21.92
3. random
    - 21.81

▼ through, back일때

1. write - through
    - 22.26
2. write - back
    - 85.98

▼ write-allocate, no-write-allocate 일때

1. write-allocate
    - 85.98
2. no-write-allocate는 write-back과 같이 사용할수 없으므로 write-through 로 적용한다면
    - 21.28

위 결과로 봤을때 가장 성능이 높은 캐시는

최대한 set수가 높고 block 도 많으며 byte_size 가 크고, write-type은 through 이고, write-allocate일때 효율이 좋다