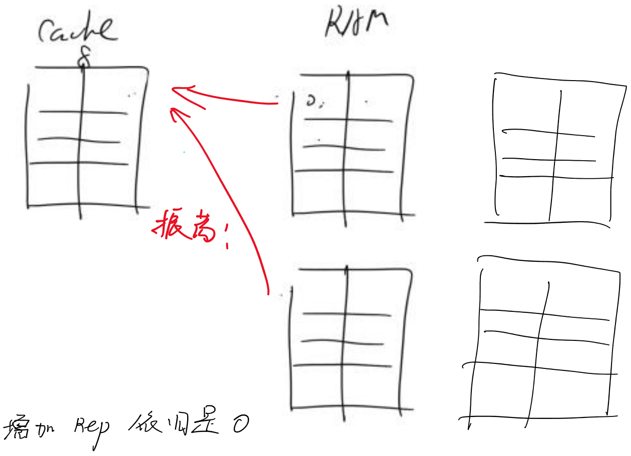
lab8

2022年4月25日 星期一 下午1:12

S1

命中 0

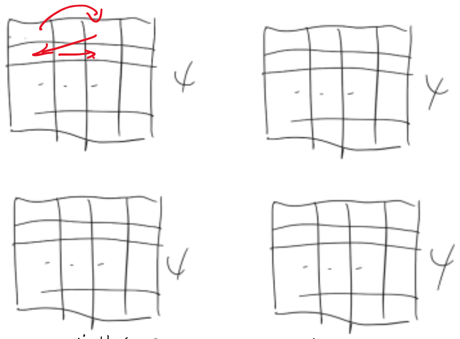


增加 Rep 依旧是 0

step size = 18

S2

读、写(必命中) → 对于一个 block, $\frac{\text{3次 hit}}{\text{4次 Access}}$
×2 (一次 miss 一次 hit)



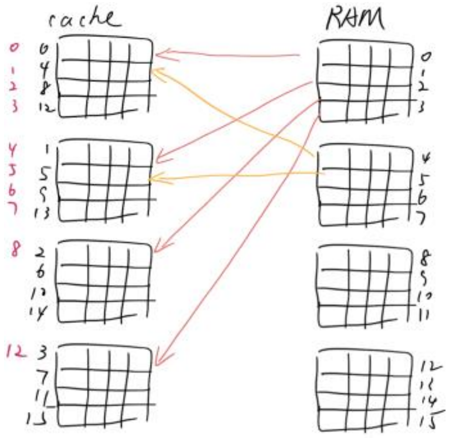
Rep 增加越接近 1，1次 Rep 全缓存
循环时间，一个 address 跑完所有循环.

S3

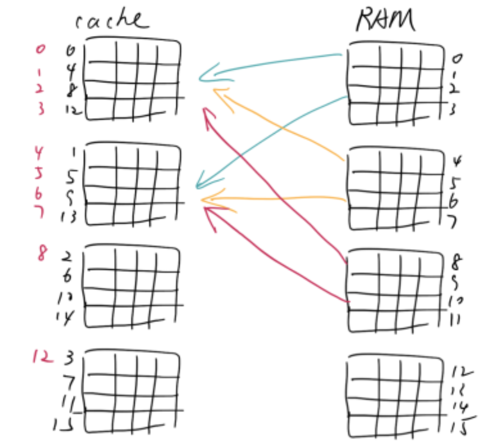
4306496104319837957	.125
43064961043198	.25

Range : 0~5

最优 (LRU)

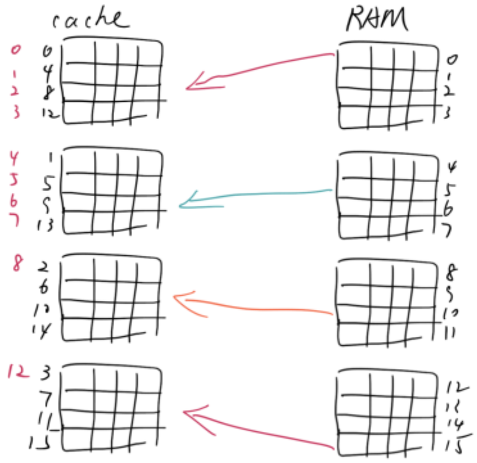


最差：全 Random 到之前的数据上.



constant hit rate:让一个组里只出一个

step size = 5



Exercise 2: Loop Ordering and Matrix Multiplication

a. Which ordering(s) perform best for 1000-by-1000 matrices?

ijk: n = 1000, 1.992 Gflop/s

ikj: n = 1000, 0.275 Gflop/s

jik: n = 1000, 1.186 Gflop/s

jki: n = 1000, 9.575 Gflop/s

kij: n = 1000, 0.289 Gflop/s

kji: n = 1000, 7.639 Gflop/s

b. Which ordering(s) perform the worst?

ijk: n = 1000, 1.992 Gflop/s

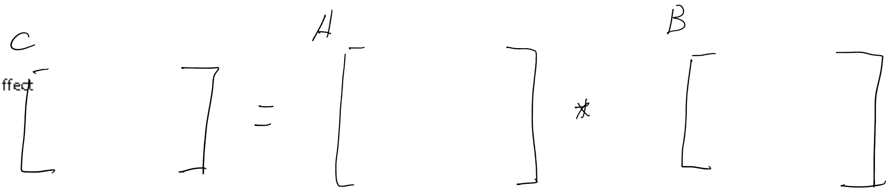
ikj: n = 1000, 0.275 Gflop/s

jik: n = 1000, 1.186 Gflop/s

jki: n = 1000, 9.575 Gflop/s

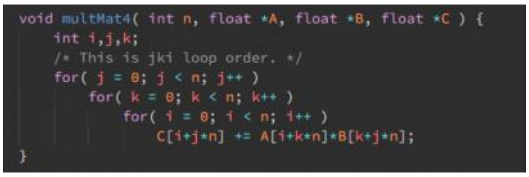
kij: n = 1000, 0.289 Gflop/s

kji: n = 1000, 7.639 Gflop/s



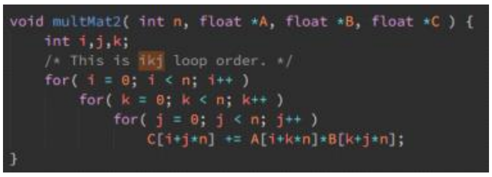
c. How does the way we stride through the matrices with respect to the innermost loop affect performance?

Best



```
void multMat4( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is jki loop order. */
    for( j = 0; j < n; j++ )
        for( k = 0; k < n; k++ )
            for( i = 0; i < n; i++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}
```

Worst



```
void multMat2( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is ikj loop order. */
    for( i = 0; i < n; i++ )
        for( k = 0; k < n; k++ )
            for( j = 0; j < n; j++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}
```

Exercise 3: Cache Blocking and Matrix Transposition

Part 1: Changing Array Sizes

./mat 100 20	Testing naive transpose: 0.011 milliseconds Testing transpose with blocking: 0.009 milliseconds
./mat 1000 20	Testing naive transpose: 3.159 milliseconds Testing transpose with blocking: 2.034 milliseconds
./mat 2000 20	Testing naive transpose: 19.73 milliseconds Testing transpose with blocking: 9.654 milliseconds
./mat 5000 20	Testing naive transpose: 163.51 milliseconds Testing transpose with blocking: 56.867 milliseconds
./mat 10000 20	Testing naive transpose: 950.274 milliseconds Testing transpose with blocking: 244.097 milliseconds

Part 2: Changing Blocksize

./transpose 10000 50	Testing naive transpose: 5870.32 milliseconds Testing transpose with blocking: 1656.2 milliseconds
./transpose 10000 100	Testing naive transpose: 5952.83 milliseconds Testing transpose with blocking: 1351.54 milliseconds
./transpose 10000 500	Testing naive transpose: 5851.51 milliseconds Testing transpose with blocking: 1259.28 milliseconds
./transpose 10000 1000	Testing naive transpose: 5910.83 milliseconds Testing transpose with blocking: 2088.36 milliseconds
./transpose 10000 5000	Testing naive transpose: 5719.15 milliseconds Testing transpose with blocking: 4961.33 milliseconds