**CACHE OPTIMISTAION :**

**MATRIX MULTIPLICATION "FOR" LOOPS :**

Assuming the cache stores in a **row major order**.

AIM : To reduce the cache miss while multiplying 2 matrixes .

**Case 1 : ijk**

for(i=0;i<=n;i++)

for(j=0;j<=n;j++)

sum=0;

for(k=0;k<=n;k++)

sum=sum + a[i][k] \* b[k][j];

c[i][j]=sum;

Matrix A elements are stored in row major .

Cache miss of A : 1 compulsory miss if column size of A <= Block size of cache. - for each row of A

Matrix b stored elements are stored in row major as well. But the column elements of B are accessed with cache miss every time.

Cache miss for B : 1 cache miss for each element .

Total miss : n (no of rows: n)

------------------------------------------------------------------------

**Case 2 :jik**

for(j=0;j<=n;j++)

for(i=0;i<=n;i++)

sum=0;

for(k=0;k<=n;k++)

sum=sum + a[i][k] \* b[k][j];

c[j][i]=sum;

Cache miss: similar to the previous case.

------------------------------------------------------------------------

**Case 3 : jki**

for(j=0;j<=n;j++)

for(k=0;k<=n;k++)

r=b[k][j];

for(i=0;i<=n;i++)

c[j][i] + = a[i][k] \* r;

Here b[k][j] is fixed .

Matrix A : Elements are stored in column major

Matrix C : Elements are stored in column major

Cache miss : 1 compulsory cache miss for each element of A and C.

(since cache is assumed to be row major )

------------------------------------------------------------------------

**Case 4 : ikj**

for(i=0;i<=n;i++)

for(k=0;k<=n;k++)

r=a[i][k];

for(j=0;j<=n;j++)

c[j][i] + = r \* b[k][j] ;

Here a[i][k] is fixed.

Matrix B : : Elements are stored in Row major

Matrix C : : Elements are stored in Row major

cache miss : 1 compulsory cache miss each for B and C when storing in row major.

-> BEST OPTIMSIATION .

------------------------------------------------------------------------

**Case 5 : kij**

for(k=0;k<=n;k++)

for(i=0;i<=n;i++)

r=a[i][k];

for(j=0;j<=n;j++)

c[j][i] + = r \* b[k][j] ;

Similar to the previous case.

------------------------------------------------------------------------

**Case 6 : kji**

for(k=0;k<=n;k++)

for(j=0;j<=n;j++)

r=b[k][j];

for(i=0;i<=n;i++)

c[j][i] + = a[i][k] \* r;

Similar to **case 3**.

------------------------------------------------------------------------

**BASED ON THE ANALYSIS OF ALL CASES**

**CASE 4 AND CASE 5**

**KIJ AND IKJ LOOP GIVES THE BEST RESULTS ASSUMING THE CACHE STORAGE IS IN ROW MAJOR.**

**1 COMULSAORY CACHE MISS WHILE LOADING EACH ROW OF MATRIX IN THE CAHCE BLOCK.**