Nikhil Ankam - 12579800

1 Theoretical analysis on matrix-matrix multiplication.

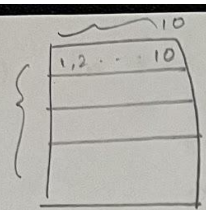
```
int dgemm(double *A, double *B, double *C, int n){
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            for (int k = 0; k < n; k++)
                C[i*n+j] += A[i*n+k] * B[k*n+j];
    return 0;
}
4 floating-point operations/cycle and 100 cycles delay
Frequency 2GHz = 2x10^9 cycles/sec
Operations:
      fetch(load)  -   n^3 x 3    -     100 cycles
      Store        -   n^3 x 1    -     100 cycles
      Sum          -   n^3 x 1    -     1/4 cycle
      Product      -   n^3 x 1    -     1/4 cycle

Total = ((3+1).100.n^3 + (1/4 + 1/4).n^3)/2x10^9 = 400.5/2 = 200.25sec
Wastage = 4.n^3/2.10^9 = 200sec

int dgemm_v(double *A, double *B, double *C, int n){
   for (int i = 0; i < n; i++)
     for (int j = 0; j < n; j++){
         double r = C[i*n+j];
         for (int k = 0; k < n; k++)
             r += A[i*n+k] * B[k*n+j];
         C[i*n+j] = r;
     }
   return 0;
}
Operations:
    Load(C)   -     n^2 x 1     -    100 cycles
       (A,B)  -     n^3 x 2     -    100 cycles
    Store (r) -     n^2 x 1     -    1/4 cycle
        (r)   -     n^3 x 1     -    1/4 cycle
        (C)   -     n^2 x 1     -    100 cycles
    Add       -     n^3 x 1     -    1/4 cycle
    Product   -     n^3 x 1     -    1/4 cycle

= 100.n^2 + 2.100.n^3 + n^2/4 + (n^3)/4 + 100.n^2 + 2/4.n^3
= 200.75(n^3 + n^2)/2x10^9 = 200.75x10^9/2x10^9 ~ 100.35sec
Wastage = 2.n^2.100 + 2.n^3.100/2x10^9 = 200.2/2 = 100.1sec
```

2)

60 

$C = C + A \times B$

Cache elements - double

LRU, one dimensional representation.

Calculate read-cache miss

1) (ijk)

```
for i=0 to n;
    for j=0 to n;
        sum = C(ixn+j)
        for k=0 to n;
            sum+= A(ixn+k) * B(kxn+j)
        C(ixn+j) = sum
```

Total reads $= n^2 + 2n^3$

n = 10000

	miss rate	iterations
C	1	n^2
A	$\frac{1}{10}$	n^3
B	1	n^3

$\Rightarrow n^2 + \frac{n^3}{10} + n^3$

$\Rightarrow n^2 + \frac{11n^3}{10}$

$$\text{Miss \%} = \frac{n^2 + \frac{11n^3}{10}}{n^2 + 2n^3} = \frac{1 + \frac{11 \times 10^4}{10}}{1 + 2 \times 10^4} = \frac{11001}{20001} = 55\%$$

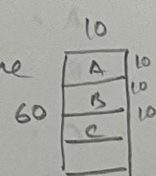
n = 10

C — $\frac{1}{10} \times 100 = 10$ (1 miss/row)

A — 10

B — 10

Since, cache



$$\text{Total} = \frac{30 \text{ miss}}{2100} = 1.4\%$$

2) (jik)

```
for j=0 to n
    for i=0 to n
        S = C(ixn+j)
        for k=0 to n
            S+= A(ixn+k) * B(kxn+j)
        C(ixn+j) = S
```

Total reads $= n^2 + 2n^3$

n = 10000

C	1	n^2
A	1/10	n^3
B	1	n^3

miss % = 55%

n = 10 total miss $= \frac{30}{2100} = 1.4\%$

(kij) for k = 0 to n
    for i = 0 to n
        r = a(i×n+k)
        for j = 0 to n
            c(i×n+j) += r × b(k×n+j)

n = 10000   A − 1 × n^2 total = $\dfrac{\frac{2n^3}{10} + n^2}{n^2 + 2n^3}$ = $\dfrac{1 + \frac{n}{5}}{1+2n}$ = $\dfrac{2001}{20001}$

B − 1/10 × n^3

C − 1/10 × n^3 = 10.%

n = 10   A − 10   = $\dfrac{30}{2100}$ = 1.4%

B − 10

C − 10

(ikj) for i = 0 to n
    for k = 0 to n
        r = a(i×n+k)
        for j = 0 to n
            c(i×n+j) += r × b(k×n+j)

n = 10000   A = 1 × n^2

B = $\frac{1}{10}$ × n^3 = 10%

C = $\frac{1}{10}$ × n^3

n = 10   30/2100 = 1.4%

(jki) for j = 0 to n
    for k = 0 to n
        r = b(k×n+j)
        for i = 0 to n
            c(i×n+j) += a(i×n+k) × r

n = 10000   B = 1 × n^2

C − 1 × n^3 = $\dfrac{n^2 + 2n^3}{n^2 + 2n^3}$ = 100%

A − 1 × n^3

n = 10 $\dfrac{30}{2100}$ = 1.4%

```
(kji)   for k=0 to n
            for j=0 to n
                r = b(k×n+j)
                for i=0 to n
                    c(i×n+j) += a(i×n+k) * r
```

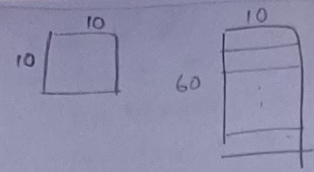
$n = 10^4$ B — 1 $\times n^2$

C — 1 $\times n^3$ =   100%

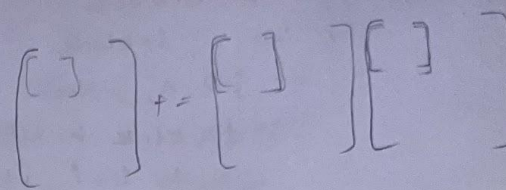
A — 1 $\times n^3$

$n = 10$ $30/2100$ = 1.4%

2.) block



LRU



(ijk)

for i=0 to n ; i+=10
    for j=0 to n , j+=10
        for k=0 to n , k+=10
            for i1=i to i+block
                for j1=j to j+block
                    for k1=k to k+block
                        $C(i1 \times n + j1) += A(i1 \times n + k1) \cdot B(k1 \times n + j1)$

n = 10,000
block = 10

Inner loop

I iteration:- for i1= 0 to 10
    for j1=0 to 10
        for k1=0 to 10
            $\underbrace{C(i1 \times n + j1)}_{10 \text{ miss}} += \underbrace{A(i1 \times n + k1)}_{10 \text{ miss}} \cdot \underbrace{B(k1 \times n + j1)}_{10 \text{ miss}}$

Total = 30 miss

Inner loop = $C = 10$ $A = \frac{n}{b} \times 10$ $B = \frac{n}{b} \times 10$

= C is still in cache because its not LRU

$$\text{Total} = \frac{\left(\frac{n}{B}\right)^2 \left[10 + \frac{2 \times 10 n}{b}\right]}{3n^3 \text{ (reads)}} \simeq \frac{20n^3}{b^3} \Big/ 3n^3 = 2/300$$

(jik)
for j=0, j+=10, n
    for i=0, i+=10, n
        for k=0, k+=10 n
            for j1=j , j+block
                for i1=i , i+block
                    for k1=k , k+block
                        $\underbrace{C(i \times n + j)}_{10} += \underbrace{A(i \times n + k1)}_{10} \cdot \underbrace{B(k \times n + j1)}_{10}$

Simillar = 2/300 %

(kij) for k=0, n, k+block
for i=0, n, i+block
for j=0, n, j+block
for k1=k, k+block
for i1=i, i+block
for j1=j, j+block
$\underbrace{C(i1\times n+j1)}_{10} += \underbrace{A(i1\times n+k1)}_{10} \cdot \underbrace{B(k1\times n+j1)}_{10}$

Inner loop $C = \frac{n}{B}\times 10$ $A = 10$ $B = \frac{n}{B}\times 10$

Total $= 2/800\,\%$ $\left[\left(\frac{n}{B}\right)^2\left(10+\frac{2n}{B}\times 10\right)\right)$

(ikj) Simillar to (kij) $= 2/800\,\%$

(jki) for j=0 to n, j+block
for k=0 to n, k+block
for i=0 to n, i+block
for j1=j to j+block
for k1=k to k+block
for i1=i to i+block
$\underbrace{C(i1\times n+j)}_{10} += \underbrace{A(i1\times n+k1)}_{10} \cdot \underbrace{B(k1\times n+j)}_{10}$

Inner loop. $C = \frac{n}{B}\times 10$, $A = \frac{n}{B}\times 10$ $B = 10$

total $= \left(\frac{2n}{B}\times 10+10\right)\left(n/B\right)^2 = 2/800\,\%$

(kji) Simillar to (jki) $= 2/800\,\%$

2 Practice on matrix-matrix multiplication

1. Code filename: HW1_1.cpp

```
n = 64
dgemm ijk GFLOPS: 1.27491, time: 0.00123371
dgemm_v ijk time: 0.000890458, error: 0
dgemm_ijk GFLOPS: 1.27833, time: 0.00123041, error: 0
dgemm_ikj GFLOPS: 0.682529, time: 0.00230446, error: 0
dgemm_jik GFLOPS: 1.40783, time: 0.00111723, error: 0
dgemm_jki GFLOPS: 1.37968, time: 0.00114002, error: 0
dgemm_kij GFLOPS: 1.46925, time: 0.00107052, error: 0
dgemm_kji GFLOPS: 1.38496, time: 0.00113567, error: 0
------------------------------
n = 128
dgemm ijk GFLOPS: 1.28775, time: 0.0097712
dgemm_v ijk time: 0.00660987, error: 0
dgemm_ijk GFLOPS: 1.28807, time: 0.00976884, error: 0
dgemm_ikj GFLOPS: 1.48245, time: 0.00848792, error: 0
dgemm_jik GFLOPS: 1.22726, time: 0.0102528, error: 0
dgemm_jki GFLOPS: 1.22359, time: 0.0102836, error: 0
dgemm_kij GFLOPS: 1.48081, time: 0.00849732, error: 0
dgemm_kji GFLOPS: 1.25487, time: 0.0100273, error: 0
------------------------------
n = 256
dgemm ijk GFLOPS: 0.804817, time: 0.125076
dgemm_v ijk time: 0.0853257, error: 0
dgemm_ijk GFLOPS: 0.844339, time: 0.119221, error: 0
dgemm_ikj GFLOPS: 1.63408, time: 0.0616025, error: 0
dgemm_jik GFLOPS: 0.848005, time: 0.118706, error: 0
dgemm_jki GFLOPS: 0.747365, time: 0.134691, error: 0
dgemm_kij GFLOPS: 1.62062, time: 0.0621139, error: 0
dgemm_kji GFLOPS: 0.746851, time: 0.134784, error: 0
------------------------------
n = 512
dgemm ijk GFLOPS: 0.829545, time: 0.97078
dgemm_v ijk time: 0.649121, error: 0
dgemm_ijk GFLOPS: 0.831726, time: 0.968235, error: 0
dgemm_ikj GFLOPS: 1.64347, time: 0.490004, error: 0
dgemm_jik GFLOPS: 0.82879, time: 0.971666, error: 0
dgemm_jki GFLOPS: 0.649057, time: 1.24073, error: 0
dgemm_kij GFLOPS: 1.63567, time: 0.492339, error: 0
```

```
dgemm_kji GFLOPS: 0.631504, time: 1.27522, error: 0
-------------------------------
n = 1024
dgemm ijk GFLOPS: 0.709617, time: 9.07878
dgemm_v ijk time: 5.57221, error: 0
dgemm_ijk GFLOPS: 0.708938, time: 9.08747, error: 0
dgemm_ikj GFLOPS: 1.64141, time: 3.92494, error: 0
dgemm_jik GFLOPS: 0.784944, time: 8.20753, error: 0
dgemm_jki GFLOPS: 0.520216, time: 12.3842, error: 0
dgemm_kij GFLOPS: 1.61572, time: 3.98735, error: 0
dgemm_kji GFLOPS: 0.49539, time: 13.0048, error: 0
-------------------------------
```

2. Code filename: HW1_0.cpp

Various register reuse along with loop orders are verified. The best results are posted in this report, all the output details can be found in HW1_0.txt

Best performance is obtained for a block of size 6x2 for loop ordering ikj.

Other considerations include ordering ijk and ikj for 2x2, 2x3, 2x4, 2x5, 3x2, 3x2, 4x2, 6x2, 3x3, 3x5

```
int dgemm_register_ikj_6x2(double *A, double *B, double *C, int n){
    register int i, j, k;
    for (i = 0; i < n; i += 6){
        for (k = 0; k < n; k += 2){
            register double a1 = A[i * n + k];
            register double a2 = A[i * n + k + 1];
            register double a3 = A[(i + 1) * n + k];
            register double a4 = A[(i + 1) * n + k + 1];
            register double a5 = A[(i + 2) * n + k];
            register double a6 = A[(i + 2) * n + k + 1];
            register double a7 = A[(i + 3) * n + k];
            register double a8 = A[(i + 3) * n + k + 1];
            register double a9 = A[(i + 4) * n + k];
            register double a10 = A[(i + 4) * n + k + 1];
            register double a11 = A[(i + 5) * n + k];
            register double a12 = A[(i + 5) * n + k + 1];

            for (j = 0; j < n; j += 2){
                register double b1 = B[k * n + j];
                register double b5 = B[k * n + j + 1];
                register double b2 = B[(k + 1)*n + j];
                register double b6 = B[(k + 1)*n + j + 1];
                // register double b3 = B[(k + 2)*n + j];
```

```
                // register double b7 = B[(k + 2)*n + j + 1];
                // register double b4 = B[(k + 3)*n + j];
                // register double b8 = B[(k + 3)*n + j + 1];

                C[i * n + j]     += a1*b1 + a2*b2;
                C[i * n + j + 1] += a1*b5 + a2*b6;

                C[(i + 1) * n + j]     += a3*b1 + a4*b2;
                C[(i + 1) * n + j + 1] += a3*b5 + a4*b6;

                C[(i + 2) * n + j]     += a5*b1 + a6*b2;
                C[(i + 2) * n + j + 1] += a5*b5 + a6*b6;

                C[(i + 3) * n + j]     += a7*b1 + a8*b2;
                C[(i + 3) * n + j + 1] += a7*b5 + a8*b6;

                C[(i + 4) * n + j]     += a9*b1 + a10*b2;
                C[(i + 4) * n + j + 1] += a9*b5 + a10*b6;

                C[(i + 5) * n + j]     += a11*b1 + a12*b2;
                C[(i + 5) * n + j + 1] += a11*b5 + a12*b6;
            }
        }
    }
    return 0;
}
```

```
dgemm_register_ikj_6x2
      time: 1.7046
      dgemm_time/time: 7.26912 times faster to dgemm
      dgemm_v_time/time: 4.82715 times faster to dgemm
      error:2.77112e-13
```

3. Code filename: HW1_2.cpp

```
dgemm execution time: 10.3318
dgemm block(2) time:10.3348
max difference = 0
dgemm block(4) time:6.3295
max difference = 0
dgemm block(8) time:5.30158
```

```
max difference = 0
dgemm block(16) time:5.18588
max difference = 0
dgemm block(32) time:5.47148
max difference = 0
dgemm block(64) time:8.09619
max difference = 0
```

The best time for n = 1024 is obtained for a block size of 16.

4. Code file_name: HW1_3.cpp

```
int dgemm_block_ikj_2x2(double *A, double *B, double *C, int n, int block){
   for (int i = 0; i < n; i += block){
       for (int k = 0; k < n; k += block){
           for (int j = 0; j < n; j += block){
               for (int i1 = i; i1 < i + block; i1 += 2){
                   for (int k1 = k; k1 < k + block; k1 += 2){
                       register double a1 = A[i1*n + k1];
                       register double a2 = A[i1*n + k1 + 1];
                       register double a3 = A[(i1 + 1) * n + k1];
                       register double a4 = A[(i1 + 1) * n + k1 + 1];
                       for (int j1 = j; j1 < j + block; j1 += 2){
                           register double b1 = B[k1*n + j1];
                           register double b3 = B[k1*n + j1 + 1];
                           register double b2 = B[(k1 + 1) * n + j1];
                           register double b4 = B[(k1 + 1) * n + j1 + 1];

                           C[i1*n + j1]             += a1*b1 + a2*b2;
                           C[i1*n + j1 + 1]         += a1*b3 + a2*b4;
                           C[(i1 + 1) * n + j1]     += a3*b1 + a4*b2;
                           C[(i1 + 1) * n + j1 + 1] += a3*b3 + a4*b4;
                       }
                   }
               }
           }
       }
   }
   return 0;
}
```

```
dgemm_v: 9.56198
n = 1024 b = 2
dgemm_block time: 8.41345     dgemm_v_time/time: 1.13651    error:0
dgemm_block_ikj_2x2     time: 2.51374     dgemm_v_time/time: 3.80389
error:1.52767e-13
-------------------------------
n = 1024 b = 4
dgemm_block time: 5.87951     dgemm_v_time/time: 1.62632    error:0
dgemm_block_ikj_2x2     time: 1.75927     dgemm_v_time/time: 5.43521
error:1.52767e-13
-------------------------------
n = 1024 b = 8
dgemm_block time: 4.8161      dgemm_v_time/time: 1.98542    error:0
dgemm_block_ikj_2x2     time: 1.74354     dgemm_v_time/time: 5.48424
error:1.52767e-13
-------------------------------
n = 1024 b = 16
dgemm_block time: 4.50618     dgemm_v_time/time: 2.12197    error:0
Dgemm_block_ikj_2x2  time: 1.50109  dgemm_v_time/time: 6.37
error:1.52767e-13
-------------------------------
n = 1024 b = 32
dgemm_block time: 4.62143     dgemm_v_time/time: 2.06905    error:0
dgemm_block_ikj_2x2     time: 1.43569     dgemm_v_time/time: 6.6602
error:1.52767e-13
-------------------------------
n = 1024 b = 64
dgemm_block time: 5.67098     dgemm_v_time/time: 1.68612    error:0
dgemm_block_ikj_2x2     time: 1.42705     dgemm_v_time/time: 6.7005
error:1.52767e-13
```

The best time is obtained for a block of size 64 with register reuse.