Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

64 kiB Cache Designing Using Jacobi Algorithm.

```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>
#include <conio.h>
#include <fstream>
#include <iostream>
#include <time.h>

#include <math.h>
#define ROTATE(a,i,j,k,l) g=a[i][j];h=a[k][l];a[i][j]=g-s*(h+g*tau);a[k][l]=h+s*(g-h*tau);
#include "nrutil.c"
#include "nrutil.h"
double *d;
int nrot;



#define cacheSize 65536
#define wB 1
#define wTA 2
#define wTNA 3
#define MAX_WAYS 16
#define MAX_LINES 16384
#define W 4

using namespace std;
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
unsigned int N[5] = { 1,2,4,8,16 };

unsigned int BL[4] = { 1,2,4,8 };

unsigned int totalLine = 0;

unsigned int lines[20];

unsigned int block = 0;

unsigned int lBits = 0;

unsigned int bBits = 0;

unsigned int tBits = 0;

unsigned int Ways = 0;

unsigned int wS = 0;


unsigned int LRU[MAX_LINES][MAX_WAYS];

bool D[MAX_LINES][MAX_WAYS];

bool V[MAX_LINES][MAX_WAYS];

unsigned int TAG[MAX_LINES][MAX_WAYS];


// write Counters


unsigned int countWriteMemory;

unsigned int countWriteBlock;

unsigned int countWriteLine;

unsigned int countWriteMiss;

unsigned int countWriteDirtyReplace;

unsigned int countWriteCache;

unsigned int countWriteThroughMemory;




// read Counters
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
unsigned int countReadMemory;

unsigned int countReadBlock;

unsigned int countReadLine;

unsigned int countReadMiss;

unsigned int countReadDirtyReplace;

unsigned int countReadCache;


//unsigned int a;

unsigned int way;

int n = 127;


void initialCache()


{


// write Counters


countWriteMemory = 0;

countWriteBlock = 0;

countWriteLine = 0;

countWriteMiss = 0;

countWriteDirtyReplace = 0;

countWriteCache = 0;

countWriteThroughMemory = 0;


// read Counters


countReadMemory = 0;
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
countReadBlock = 0;

countReadLine = 0;

countReadMiss = 0;

countReadDirtyReplace = 0;

countReadCache = 0;


for (int clear_line = 0; clear_line<MAX_LINES; clear_line++)

{

for (int clear_way = 0; clear_way < MAX_WAYS; clear_way++)

{

//LRU[clear_line][clear_way] = 0;

TAG[clear_line][clear_way] = 0;

V[clear_line][clear_way] = 0;

D[clear_line][clear_way] = 0;

}

}


}



void lru()

{

bool V = 0;

bool D = 0;

unsigned int way;

unsigned int line;


for (line = 0;line<MAX_LINES;line++)

{
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
for (way = 0;way<MAX_WAYS;way++)
{
LRU[line][way] = way;
}
}
}

void lineValues()
{
int x, y, z = 0;
for (x = 0; x < 4;x++) // BL
{
for (y = 0; y < 5; y++)     // number of ways
{
lines[z] = (cacheSize / (N[y] * BL[x] * W));
z++;
}
}
}

bool isMiss(unsigned int address)
{
unsigned int line, tag = 0;
unsigned int w = 0;
bool miss = true;

line = (address >> bBits) & (unsigned int(pow(2, lBits) - 1));
tag = (address >> (lBits + bBits)) & (unsigned int(pow(2, (lBits + bBits)) - 1));
for(w = 0; w <= Ways-1; w++)
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
{
if ((TAG[line][w] == tag) && (V[line][w] == 1))
{
miss = false;
break;
}
}
return miss;
}

int getLine(unsigned int address)
{
unsigned int line = 0;
line = (address >> bBits) & (unsigned int(pow(2, lBits) - 1));
return line;
}

int getTag(unsigned int address)
{
unsigned int tag = 0;
tag =(address >> (lBits + bBits)) & (unsigned int(pow(2, tBits) - 1));
return tag;
}

int getLRUWay(unsigned int lru_line1)
{
unsigned int  lru_way1 = 0;
unsigned int lru_value = LRU[lru_line1][0];
unsigned int temp_lru = 0;
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
for (lru_way1 = 1; lru_way1 <= Ways - 1; lru_way1++)
{
if (lru_value < LRU[lru_line1][lru_way1])
{
lru_value = LRU[lru_line1][lru_way1];
temp_lru = lru_way1;
}
}
return temp_lru;
}
```

```
bool isDirty(unsigned int dirty_line, unsigned int dirty_way)
{
if (D[dirty_line][dirty_way] == true)
{
return true;
}
else if (D[dirty_line][dirty_way] == false)
{
return false;
}

}
```

```
void writeBlock(unsigned int w_line, unsigned int w_way)
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
{
//increment the counter for wite to memory
countWriteBlock++;
}

/*void invalidate(unsigned int inv_line, unsigned int inv_way)
{
V[inv_line][inv_way] = 0;
}*/

void readBlock(unsigned int rd_line, unsigned int rd_way)
{
//increment counter to read from memory
countReadBlock++;

}

void validate(unsigned int v_line, unsigned int v_way)
{
V[v_line][v_way] = 1;
}

void updateLRU(unsigned update_address)
{
unsigned int update_line, update_way, u_way, tag, n, n1, temp_n = 0;
tag = getTag(update_address);
update_line = getLine(update_address);
if (!(isMiss(update_address)))
{
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
for (n = 0; n <= Ways - 1; n++)
{
if (TAG[update_line][n] == tag )
{
temp_n = n;
break;
//LRU[update_line][n] = 0;

}
}
}
else if(isMiss(update_address))
{
temp_n = way;
}

for (n1 = 0; n1 <= Ways - 1; n1++)
{
if (LRU[update_line][n1] < LRU[update_line][temp_n])
{
LRU[update_line][n1]++;
}
}
LRU[update_line][temp_n] = 0;
}

void readCache(unsigned int cache_address)
{
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
//increment the counter for cache read
countReadCache++;
}

void readLine(unsigned int rl_add)
{
countReadLine++;
unsigned int rl_line = 0,rl_tag = 0;

if (isMiss(rl_add))
{
countReadMiss++;
rl_line = getLine(rl_add);
rl_tag = getTag(rl_add);
way = getLRUWay(rl_line);

if (isDirty(rl_line, way))
{
countReadDirtyReplace++;
writeBlock(rl_line, way);          //Write back to memory
D[rl_line][way] = false;           //clear dirty bit
}

TAG[rl_line][way] = rl_tag;
readBlock(rl_line, way);
validate(rl_line, way);

}
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
updateLRU(rl_add);

readCache(rl_add);


}



void readMemory(void *add32, unsigned int size)

{

countReadMemory++;

unsigned int rm_address = 0, rm_line, i = 0;

int oldLine = -1;


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

{

rm_address = (unsigned int)add32 + i;

rm_line = getLine(rm_address);

if (rm_line != oldLine)

{

oldLine = rm_line;

readLine(rm_address);

}


}

rm_line = 0;

}


void writeCache(unsigned int cache_address)

{

//inc cntr for write cache
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
countWriteCache++;


}


void writeLine(unsigned int wl_add)
{
countWriteLine++;
unsigned int wl_line =0, wl_tag = 0;
//unsigned int wl_add = (unsigned int)add;
//bool hit = isHit(wl_add);
//bool miss = ~hit;
wl_line = getLine(wl_add);
wl_tag = getTag(wl_add);
if (isMiss(wl_add))
{
countWriteMiss++;
if ((wS == wB) || (wS == wTA))
{

//wl_line = getLine(wl_add);
way = getLRUWay(wl_line);



if (isDirty(wl_line, way))
{
countWriteDirtyReplace++;
writeBlock(wl_line, way);
D[wl_line][way] = false;                    //clear dirty bit
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
}

TAG[wl_line][way] = wl_tag;
readBlock(wl_line, way);
validate(wl_line, way);



}
}

if (!(isMiss(wl_add)) || wS == wB || wS == wTA)
{
if (wS == wB)
{
wl_line = getLine(wl_add);
D[wl_line][way] = true;
}


writeCache(wl_add);
updateLRU(wl_add);
}



}

void writeThruMemory(unsigned int address)
{
// inc cntr for write thru mem
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
countWriteThroughMemory++;
}

void writeMemory(void *add32, unsigned int size)
{
countWriteMemory++;
unsigned int wm_address, wm_line, add1, line_bits, i = 0;
int oldLine = -1;
int oldlineBits = -1;
for (i = 0; i < size; i++)
{

wm_address = (unsigned int)add32 + 4;
wm_line = getLine(wm_address);
if (wm_line != oldLine)
{
oldLine = wm_line;
writeLine(wm_address);
}



if ((wS == wTA) || (wS == wTNA))
{
add1 = (unsigned int)add32 + i;
line_bits = (add1 >> 2) & (unsigned int(pow(2, 2)) - 1);          //since write thru memory is 32-bit bounded
if (line_bits != oldlineBits)
{
oldlineBits = line_bits;
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
writeThruMemory(add1);
}


}

/*wm_address = (unsigned int)add32 + i;
wm_line = getLine(wm_address);
if (wm_line != oldLine)
{
oldLine = wm_line;
writeLine(wm_address);
}*/
}
}


void testCase()
{
int c;
int i = 0;
int        j = 0;

__declspec(align(4)) double **a = dmatrix(0, n, 0, n);
__declspec(align(4)) double **v = dmatrix(0, n, 0, n);
__declspec(align(4)) double *d = dvector(0, n);

for (i = 0; i < 127; i++)
{
for (j = 0; j < 127; j++)
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
{
if (i == j)
a[i][j] = rand() % 32 + 1;
else
a[i][j] = a[j][i] = rand() % 32 + 1;

}
}
/*for (i = 0; i < 127; i++)
{
for (j = 0; j < 127; j++)
{
printf("%lf\t", a[i][j]);
}
printf("\n");
}*/

jacobi(a, n, d, v, &nrot);

/*__declspec(align(4))  double x[16384];
__declspec(align(4))  double z[16384];
unsigned int a,i;

for (i = 0; i < 2; i++)
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
{
for (a = 0; a < 16384; a++)
{
x[a] = 0;//x[a] + 1;

//writeMemory(&x[a], sizeof(x[a]));
readMemory(&x[a], sizeof(x[a]));
writeMemory(&x[a], sizeof(x[a]));
writeMemory(&x[a], sizeof(x[a]));
readMemory(&x[a], sizeof(x[a]));

readMemory(&z[a], sizeof(z[a]));
writeMemory(&z[a], sizeof(z[a]));
writeMemory(&z[a], sizeof(z[a]));
readMemory(&z[a], sizeof(z[a]));

}
}*/



}



void jacobi(double **a, int n, double d[], double **v, int *nrot)
//Computes all eigenvalues and eigenvectors of a real symmetric matrix a[1..n][1..n]. On
//output, elements of a above the diagonal are destroyed. d[1..n] returns the eigenvalues of a.
//v[1..n][1..n] is a matrix whose columns contain, on output, the normalized eigenvectors of
//a. nrot returns the number of Jacobi rotations that were required.
{
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
{
int j, iq, ip, i;
double tresh, theta, tau, t, sm, s, h, g, c, *b, *z;

b = vector(1, n);
writeMemory(&b, sizeof(b));

z = vector(1, n);
writeMemory(&z, sizeof(z));

writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
for (ip = 1;ip <= n;ip++) { //Initialize to the identity matrix.

writeMemory(&iq, sizeof(iq));
readMemory(&iq, sizeof(iq));
readMemory(&n, sizeof(n));
for (iq = 1;iq <= n;iq++)
{

v[ip][iq] = 0.0;
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
writeMemory(&v[ip][iq], sizeof(v[ip][iq]));
v[ip][ip] = 1.0;
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
writeMemory(&v[ip][ip], sizeof(v[ip][ip]));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
readMemory(&iq, sizeof(iq));
readMemory(&n, sizeof(n));
writeMemory(&iq, sizeof(iq));
readMemory(&iq, sizeof(iq));

}
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));

}


writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
for (ip = 1;ip <= n;ip++)
{                                   //Initialize b and d to the diagonal

b[ip] = d[ip] = a[ip][ip];                  //of a.
readMemory(&ip, sizeof(ip));
readMemory(&a[ip], sizeof(a[ip]));
writeMemory(&d[ip], sizeof(d[ip]));
readMemory(&d[ip], sizeof(d[ip]));
writeMemory(&b[ip], sizeof(b[ip]));
z[ip] = 0.0;                            //This vector will accumulate terms
readMemory(&ip, sizeof(ip));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
writeMemory(&z[ip], sizeof(z[ip]));
//of the form tapq as in equation (11.1.14).
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
}
*nrot = 0;
writeMemory(&nrot, sizeof(nrot));

writeMemory(&i, sizeof(i));
readMemory(&i, sizeof(i));
for (i = 1;i <= 50;i++)                    // here it was 50 but i changed it to l=50
{
sm = 0.0;
writeMemory(&sm, sizeof(sm));

writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
for (ip = 1;ip <= n - 1;ip++)
{                                  // Sum off-diagonal elements.

readMemory(&ip, sizeof(ip));
writeMemory(&iq, sizeof(iq));
readMemory(&iq, sizeof(iq));
readMemory(&n, sizeof(n));
for (iq = ip + 1;iq <= n;iq++)
{
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
sm += fabs(a[ip][iq]);
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
readMemory(&a[ip][iq], sizeof(a[ip][iq]));
writeMemory(&sm, sizeof(sm));

readMemory(&iq, sizeof(iq));
readMemory(&n, sizeof(n));
writeMemory(&iq, sizeof(iq));
readMemory(&iq, sizeof(iq));
}

readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));

}
readMemory(&sm, sizeof(sm));
if (sm == 0.0)

{                                       //The normal return, which relies on quadratic convergence to
machine underflow.

free_dvector(z, 1, n);
readMemory(&n, sizeof(n));
readMemory(&z, sizeof(z));
writeMemory(&z, sizeof(z));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
free_dvector(b, 1, n);
readMemory(&n, sizeof(n));
readMemory(&b, sizeof(b));
writeMemory(&b, sizeof(b));

return;                                    // check for the return command may be related to bx lr
// readMemory(&lr, sizeof(lr));



}

readMemory(&i, sizeof(i));

if (i < 4)
{
tresh = 0.2*sm / (n*n);                //...on the first three sweeps.
readMemory(&n, sizeof(n));             // shoud we take it twice ?
readMemory(&sm, sizeof(sm));
writeMemory(&tresh, sizeof(tresh));
}
else

tresh = 0.0;                           //...thereafter.
writeMemory(&tresh, sizeof(tresh));

writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
for (ip = 1;ip <= n - 1;ip++)
{
readMemory(&ip, sizeof(ip));
writeMemory(&iq, sizeof(iq));
readMemory(&iq, sizeof(iq));
readMemory(&n, sizeof(n));
for (iq = ip + 1;iq <= n;iq++)
{
g = 100.0*fabs(a[ip][iq]);
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
readMemory(&a[ip][iq], sizeof(a[ip][iq]));
writeMemory(&g, sizeof(g));
// After four sweeps, skip the rotation if the off-diagonal element is small.

readMemory(&i, sizeof(i));
readMemory(&g, sizeof(g));
readMemory(&ip, sizeof(ip));
readMemory(&d[ip], sizeof(d[ip]));    // do we have to readMemory one more time here?
readMemory(&iq, sizeof(iq));
readMemory(&d[iq], sizeof(d[iq]));
readMemory(&iq, sizeof(iq));
if (i > 4 && (float)(fabs(d[ip]) + g) == (float)fabs(d[ip]) && (float)(fabs(d[iq]) + g) == (float)fabs(d[iq]))
{
a[ip][iq] = 0.0;
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
writeMemory(&a[ip][iq], sizeof(a[ip][iq]));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
readMemory(&a[ip][iq], sizeof(a[ip][iq]));
readMemory(&tresh, sizeof(tresh));
}
else if (fabs(a[ip][iq]) > tresh)
{

h = d[iq] - d[ip];
readMemory(&ip, sizeof(ip));
readMemory(&d[ip], sizeof(d[ip]));
readMemory(&iq, sizeof(iq));
readMemory(&d[iq], sizeof(d[iq]));
writeMemory(&h, sizeof(h));

readMemory(&g, sizeof(g));
readMemory(&h, sizeof(h)); // doubt?
if ((float)(fabs(h) + g) == (float)fabs(h))
{
t = (a[ip][iq]) / h;                    //t = 1/(2θ)
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
readMemory(&a[ip][iq], sizeof(a[ip][iq]));
readMemory(&h, sizeof(h));
writeMemory(&t, sizeof(t));
}

else
{
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
theta = 0.5*h / (a[ip][iq]);        // Equation (11.1.10).
readMemory(&ip, sizeof(ip));
readMemory(&iq, sizeof(iq));
readMemory(&a[ip][iq], sizeof(a[ip][iq]));
readMemory(&h, sizeof(h));
writeMemory(&theta, sizeof(theta));



t = 1.0 / (fabs(theta) + sqrt(1.0 + theta*theta));
readMemory(&theta, sizeof(theta));
writeMemory(&t, sizeof(t));

readMemory(&theta, sizeof(theta));
if (theta < 0.0)
t = -t;
readMemory(&t, sizeof(t));
writeMemory(&t, sizeof(t));
}



c = 1.0 / sqrt(1 + t*t);
readMemory(&t, sizeof(t));
writeMemory(&c, sizeof(c));

s = t*c;
readMemory(&t, sizeof(t));
readMemory(&c, sizeof(c));
writeMemory(&s, sizeof(s));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
tau = s / (1.0 + c);

readMemory(&s, sizeof(s));

readMemory(&c, sizeof(c));

writeMemory(&tau, sizeof(tau));


h = t*a[ip][iq];

readMemory(&ip, sizeof(ip));

readMemory(&iq, sizeof(iq));

readMemory(&a[ip][iq], sizeof(a[ip][iq]));

writeMemory(&h, sizeof(h));


z[ip] -= h;

readMemory(&h, sizeof(h));

readMemory(&ip, sizeof(ip));

writeMemory(&z[ip], sizeof(z[ip]));


z[iq] += h;

readMemory(&h, sizeof(h));

readMemory(&iq, sizeof(iq));

writeMemory(&z[iq], sizeof(z[iq]));


d[ip] -= h;

readMemory(&h, sizeof(h));

readMemory(&ip, sizeof(ip));

writeMemory(&d[ip], sizeof(d[ip]));


d[iq] += h;

readMemory(&h, sizeof(h));

readMemory(&iq, sizeof(iq));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
writeMemory(&d[iq], sizeof(d[iq]));


a[ip][iq] = 0.0;
readMemory(&iq, sizeof(iq));
readMemory(&ip, sizeof(ip));
writeMemory(&a[ip][iq], sizeof(a[ip][iq]));


writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
readMemory(&ip, sizeof(ip));
for (j = 1;j <= ip - 1;j++)
{                                       //Case of rotations 1 ≤ j<p.
ROTATE(a, j, ip, j, iq)
readMemory(&a, sizeof(a));
readMemory(&j, sizeof(j));
readMemory(&ip, sizeof(ip));
readMemory(&j, sizeof(j));
readMemory(&iq, sizeof(iq));


readMemory(&j, sizeof(j));
readMemory(&ip, sizeof(ip));
writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));


}


readMemory(&ip, sizeof(ip));
writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
readMemory(&iq, sizeof(iq));
for (j = ip + 1;j <= iq - 1;j++)
{                                   //Case of rotations p<j<q.
ROTATE(a, ip, j, j, iq)
readMemory(&a, sizeof(a));
readMemory(&ip, sizeof(ip));
readMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
readMemory(&iq, sizeof(iq));


readMemory(&j, sizeof(j));
readMemory(&iq, sizeof(iq));
writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
}

readMemory(&iq, sizeof(iq));
writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
readMemory(&n, sizeof(n));
for (j = iq + 1;j <= n;j++)
{                                   //Case of rotations q<j ≤ n.
ROTATE(a, ip, j, iq, j)
readMemory(&a, sizeof(a));
readMemory(&ip, sizeof(ip));
readMemory(&j, sizeof(j));
readMemory(&iq, sizeof(iq));
readMemory(&j, sizeof(j));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
readMemory(&j, sizeof(j));
readMemory(&n, sizeof(n));
writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
}

writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
readMemory(&n, sizeof(n));
for (j = 1;j <= n;j++)
{
ROTATE(v, j, ip, j, iq)
readMemory(&v, sizeof(v));
readMemory(&j, sizeof(j));
readMemory(&ip, sizeof(ip));
readMemory(&j, sizeof(j));
readMemory(&iq, sizeof(iq));

readMemory(&j, sizeof(j));
readMemory(&n, sizeof(n));
writeMemory(&j, sizeof(j));
readMemory(&j, sizeof(j));
}

++(*nrot);
readMemory(&nrot, sizeof(nrot));
writeMemory(&nrot,sizeof(nrot));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
}

readMemory(&iq, sizeof(iq));
readMemory(&n, sizeof(n));
writeMemory(&iq, sizeof(iq));
readMemory(&iq, sizeof(iq));
}

readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
}

writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
for (ip = 1;ip <= n;ip++)
{

b[ip] += z[ip];
readMemory(&ip, sizeof(ip));
readMemory(&z[ip], sizeof(z[ip]));
writeMemory(&b[ip], sizeof(b[ip]));

d[ip] = b[ip];                //Update d with the sum of tapq,
readMemory(&ip, sizeof(ip));
readMemory(&b[ip], sizeof(b[ip]));
writeMemory(&d[ip], sizeof(d[ip]));
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
z[ip] = 0.0;                    //and reinitialize z.
readMemory(&ip, sizeof(ip));
writeMemory(&z[ip], sizeof(z[ip]));

readMemory(&ip, sizeof(ip));
readMemory(&n, sizeof(n));
writeMemory(&ip, sizeof(ip));
readMemory(&ip, sizeof(ip));
}
readMemory(&ip, sizeof(i));
writeMemory(&ip, sizeof(i));
readMemory(&ip, sizeof(i));
}

nrerror("Too many iterations in routine jacobi"); // doubt ?

}



void main()
{
//  ofstream outfile;
//   outfile.open("DATA.txt");

ofstream outfile;
outfile.open("Cache_Analysis.txt");
unsigned int nways = 0;
unsigned int burst = 0;
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
unsigned int k = 0;

lru();

lineValues();


/*printf("Cache Analysis\n");

printf("WRITE STRATEGY: 0 : Write back, 1 : Write Through Allocate, 2 : Write Through Non-Allocate\n");

printf("Burst Length = {1, 2, 4, 8}\n");

printf("Number of Ways = {1, 2, 4, 8, 16}\n");

printf("Testing for 60 Combination..\n");*/


for (wS = 1;wS<4; wS++)

{

if (wS == 1)

printf("write Strategy : Write Back \n");


if (wS == 2)

printf("write Strategy : Write Through Allocate \n");


if (wS == 3)

printf("write Strategy : Write Through Non Allocate \n");


for (burst = 0; burst<4;burst++)

{


for (nways = 0; nways<5; nways++)

{

totalLine = lines[k];

block = BL[burst];

Ways = N[nways];
```

Name: Dhaval Padia
Email: dhavalhimanshub.padia@mavs.uta.edu

```
lBits = log2(lines[k]);

bBits = log2(W * BL[burst]);

tBits = (32 - lBits - bBits);

k++;

if (k == 20)

k = 0;

testCase();

outfile << wS << "\t" << totalLine << "\t" << block << "\t" << Ways << "     \t"

<< countReadMemory << "\t" <<"\t"<< countReadBlock<<"\t"<< countReadLine << "\t" <<
countReadMiss << "\t" << countReadDirtyReplace << "\t" << countReadCache << "         \t"

<< countWriteMemory <<"\t"<<countWriteBlock<<"\t"<<countWriteLine << "\t" << countWriteMiss <<
"\t" << countWriteDirtyReplace << "\t" << countWriteCache <<"\t"<<countWriteThroughMemory<<
endl;


initialCache();

}


}

}


printf("\n\r Test Case Complete!!...");

outfile.close();

getchar();

return;

}
```