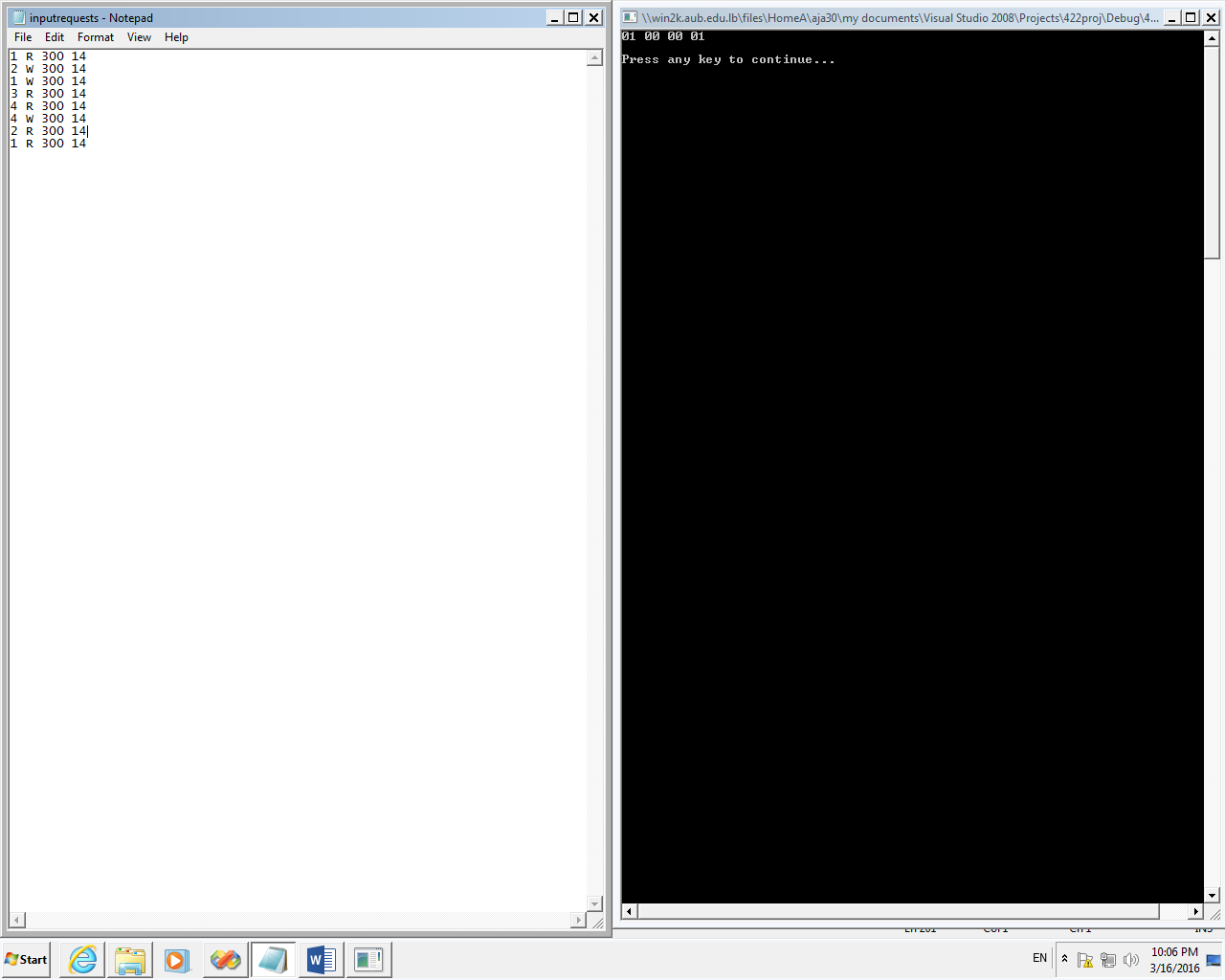
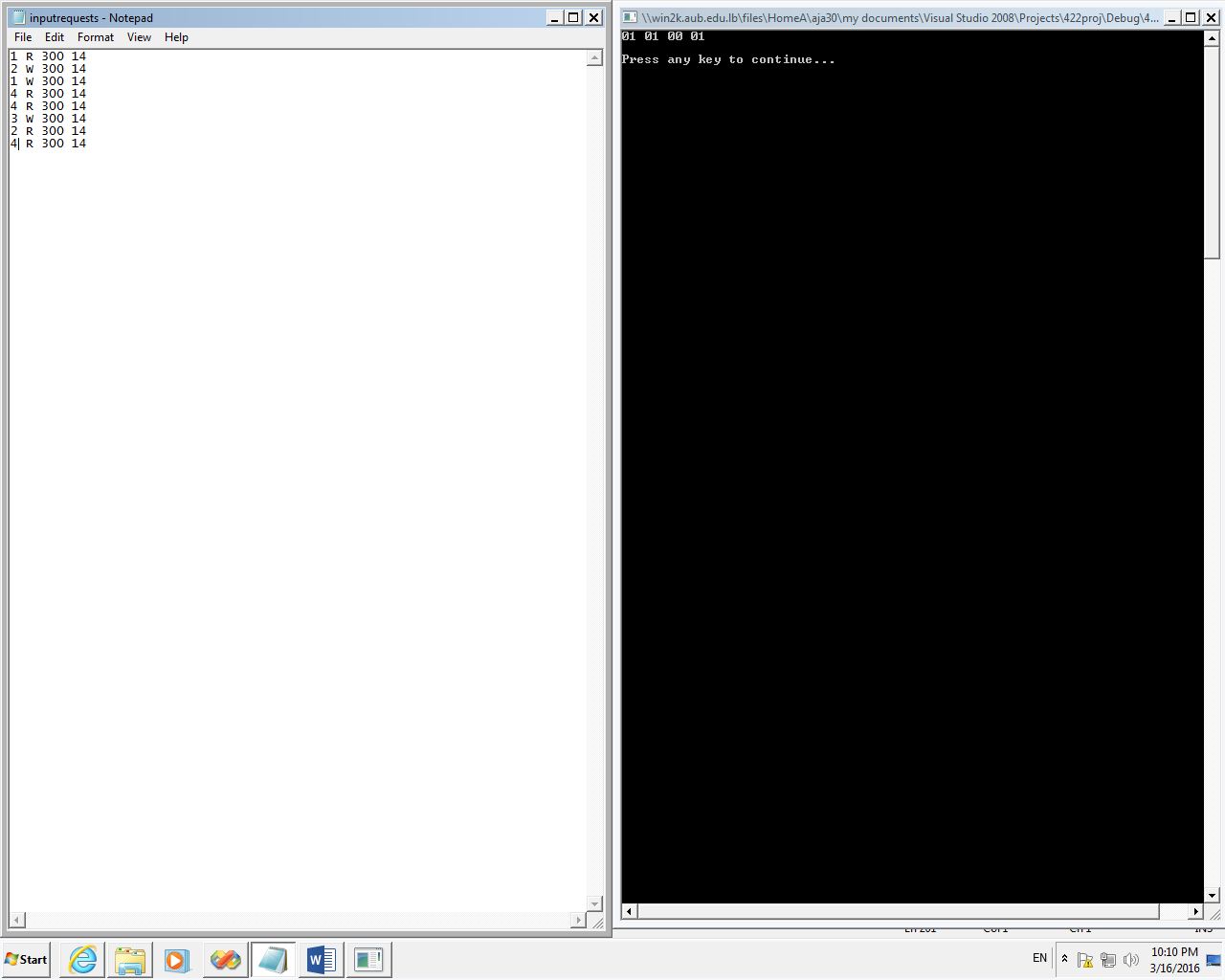
**Output:**

Note: In the below output, 01 00 00 01 means nodes 0 and 3 are shared and nodes 1 and 2 are invalid.

Trying at index 300 tag 4 for all caches, the following reads and writes.



Another output:



**CODE:**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include <cmath>

#include <omp.h>

#include <sstream>

#define chunk 1

using namespace std;

//for simplicity: tag and index are integers

class Cache

{

//the index of each array is the actual index in the cache 0->499

private:

public:

std::string state[500];

std::string state2[500];

//2-way set associative, so each index corresponds to 2 tags

int tags1[500]; //tag of first block in set @ index which is the array index

int tags2[500]; //second block

Cache()

{

for(int j = 0;j <500;j++)

{

tags1[j] = 0; //intialization for invalid blocks in cache...

tags2[j] = 0;

}

for (int i = 0; i < 500; i++)

{

state[i] = "00";

state2[i] = "00"; // initially all blocks at index i are invalid

}

}

~Cache()

{

}

};

string firstLine(string s, string &sminus);

string substr(string s, int i);

int strtonum(string s);

void main()

{

int clock\_cycles = 50; //is 16

int tid;

string bus ="0 0 0 0"; //takes the request one at a time ..

string requests[4]; // example: requests[0] is the string of requests of thread 1

Cache \*caches = new Cache[4]; //each cache is by its index

ifstream in;

in.open("inputrequests.txt");

if (in.is\_open())

{

while (!in.eof()) //request: cache\_number W/R index tag

{

string line;

getline(in, line);

char c = line[0];

int id = (int)c - '0';

requests[id-1] += line +"\n"; //check if need newline character for function

}

}

in.close();

int n = 0; //"priority" specifies which thread is requesting to bus

omp\_set\_num\_threads(4);

//initialize some blocks in some caches as a test for the program state machine

caches[0].tags1[300] = 14;

caches[1].tags1[300] = 14;

caches[2].tags1[300] = 14;

caches[3].tags1[300] = 14;

caches[1].tags2[4] = 211;

caches[2].tags1[4] = 123;

caches[1].tags2[7] = 234;

caches[3].tags1[8] = 345;

caches[3].tags2[10] = 352;

int i = 0;

int flagk = 0;

//each thread has a priority to send request on bus

#pragma omp parallel default(shared) private(i,tid/\*priority\*/)

{

tid = omp\_get\_thread\_num();

//#pragma omp for schedule(static,chunk)

for (i = 0; i < clock\_cycles; i++) //in each clock cycle a prioritized thread handles a message

{

//if the node has a request

if (tid == n)

{

string sminusfirstline;

string current\_req;

current\_req = firstLine(requests[tid],sminusfirstline);

requests[tid] = sminusfirstline; //removing every first line after done with request

if (current\_req !="" && current\_req!= "\n")

{

flagk = 1;

//example: 1 R 300 14

char inst = current\_req[2];

string indstr = substr(current\_req,2);

string tagstr = substr(current\_req,3);

int index = strtonum(indstr);

int tag = strtonum(tagstr);

//we are in caches[tid]

//note that array indeces are the actual index in cache attributes.

if (inst == 'R')

{

if (tag == caches[tid].tags1[index])

{

if (caches[tid].state[index] == "00")

{

caches[tid].state[index] = "01"; //read miss , memory write back

bus = "10 01 " + indstr +" " +tagstr; //read miss

}

}

else if(tag == caches[tid].tags2[index])

{

if (caches[tid].state2[index] == "00")

{

caches[tid].state2[index] = "01";

bus = "10 01 " + indstr +" " +tagstr;

}

}

}

else if (inst == 'W')

{

if (tag == caches[tid].tags1[index])

{

if (caches[tid].state[index] == "00" || caches[tid].state[index] == "01") //write miss

{

caches[tid].state[index] = "10";

bus = "11 00 "+ indstr +" " +tagstr;

}

}

else if (tag == caches[tid].tags2[index])

{

if (caches[tid].state2[index] == "00" || caches[tid].state[index] == "01")

{

caches[tid].state2[index] = "10";

bus = "11 00 "+ indstr +" " +tagstr; //01>00 and 10>00 invalidate everything

}

}

}

}

else { flagk = 0;}

}

#pragma omp barrier

if(n!= tid && flagk == 1) //flagk is to ensure that if request is empty for current priority cache, the others dont read bus

{

int index=strtonum(substr(bus, 2));

int tag=strtonum(substr(bus, 3));

string flag=substr(bus, 0);

string state=substr(bus, 1);

if (caches[tid].tags1[index] == tag)

{

if (caches[tid].state[index] == flag || flag =="11")

{

caches[tid].state[index] = state;

}

}

if (caches[tid].tags2[index] == tag || flag =="11")

{

if (caches[tid].state2[index] == flag)

{

caches[tid].state2[index] = state;

}

}

}

#pragma omp barrier

#pragma omp single

{

n = (n+1)%4;

}

}

}

string stateExample = caches[0].state[300]; //state of block in cache 0 and index 300 at specific tag 14

string stateExample1 = caches[1].state[300];

string stateExample2 = caches[2].state[300];

string stateExample3 = caches[3].state[300];

cout<<stateExample<<" "<<stateExample1<<" "<<stateExample2<<" "<<stateExample3<<"\n";

cout << "\nPress any key to continue...";

}

string firstLine(string s, string &sminus)

{

string temp = "";

int len = s.length();

for (int i = 0; i < len ; i++)

{

temp += s[i];

if (s[i] == '\n')

{

for (int j = i+1 ;j< len; j++)

{

sminus += s[j];

}

break;

}

}

return temp;

}

string substr(string s, int i) //s= "1 R 300 14" //returns 300 if arguments are(s,2)

{

string temp = "";

int len = s.length();

int counter = 0;

for(int k=0;k<len;k++)

{

if(s[k] == ' ')

{

counter++;

}

if (counter == i+1)

{

break;

}

if (counter == i)

{

if(s[k] != ' ')

temp+= s[k];

}

}

return temp;

}

int strtonum(string s)

{

int kk = 0;

stringstream ss(s);

ss >> kk;

return kk;

}