## Direct Mapped

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Computer Architecture 3 Formal Element Cache Simulator

Direct Mapped Cache

32-bit CPU Addr, 8-bit data bus, 4 Byte data lines, 64kB data cache

Written in VS Studio 2017 by Jack Harding

08/11/2018

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

#include "pch.h"

using namespace std;

struct cacheEntry{ // what will be in each cahe entry

unsigned short upprAddr; // creating variable for upper address

bool validFlag; // valdid flag indicates whether or not cache entry is valid

bool LRUFlag; // least recently used flag indicates if the cache entry was the last to be used

char data[4]; // array for storing data

};

cacheEntry way0[16384]; // creating way, 2^14 entries

string addrString; // string line

ifstream myAddr("testAddresses.txt"); // file for addresses

unsigned int CPUAddr; // 32-bit CPU addr

unsigned char byteNo; // 8-bit No.

unsigned short CPUUppr, setNo, hits = 0, misses = 0; // tag addr, which elemnt in way, hit, miss counter

int main()

{

cout << "Direct Mapped Cache Simulator\n\n";

for (int i = 0; i <= 16384; i++) { //initialising LRU flags and validflags

way0[i].validFlag = 0;

way0[i].LRUFlag = 1; // not used in the direct mapped

}

if (myAddr.is\_open()) {

while (getline(myAddr, addrString)){

stringstream tempS(addrString); // reading first line of addr file

tempS >> hex >> CPUAddr; // string to hex

CPUUppr = (CPUAddr & 0xffff0000) >> 16; // bit shifting by 16, anded to get CPU upper addr

setNo = (CPUAddr & 0x0000fffc) >> 2; // and with fffc, shift by 2 bits

byteNo = CPUAddr & 3; // and CPU addr with 3 to show last 2 bits

if ((way0[setNo].upprAddr == CPUUppr) && (way0[setNo].validFlag == 1)) { // hit

string\* pByte = (string\*)byteNo; // typecasts the byteNo to printed

cout << "Full Addr: 0x" << hex << CPUAddr;

cout << " Hit: 0x" << hex << CPUUppr << setNo << " " << pByte << endl;// formatting

hits++; // incrementing hit counter everytime a hit occurs

}

else { //miss

string\* pByte = (string\*)byteNo;

cout << "Full Addr: 0x" << hex << CPUAddr;

cout << " Miss: 0x" << hex << CPUUppr << setNo << " " << pByte << endl;

way0[setNo].upprAddr = CPUUppr; // storing CPUUppr into loc at setNo

way0[setNo].validFlag = 1; // is valid

misses++;

}

}

myAddr.close(); // closing file

}

else // file doesn't open error

cout << "\nFile Open Failed\n";

cout << "\nTotal Hits: " << dec << hits << endl;

cout << "Total Misses: " << dec << misses << endl; // must be dec for values over 9

system("pause");

return 0;

}