Michael Gibson
ECE 356
Lab 2

Cache Simulator

The cache simulator file contains 3 parts. The program itself (lab2.cpp), the address.txt file, and this document. Make sure to save the address.txt and lab2.cpp are in the same file when running the program. The program is in c++, and compiled with g++ -o lab2 lab2.cpp. When running the program, you can add up to 5 parameters

The parameters are in this order : block size, number of blocks, hit time (ns), miss time (ns), associativity
Default values are as follow:
Block size = 16
Number of blocks = 128
Hit time = 1
Miss time = 1
Associativity = 1

Associativity must be 1, as it is only directly mapped. The program will not allow you to pick any other value.

The program will use the addresses.txt file and read from its addresses. If addresses are to be replaced in said file, it must contain a 0x at the start of each address, and must end with and endl.

The program works by taking in the parameters and settings ints/floats accordingly.

```
int main(int argc, char *argv[]){
    int block_size = 16;//size of each block
    int num_of_blocks = 128;//number of total blocks
    int associative = 1;
    float hit_time = 1;//time it takes for a hit
    float miss_time = 1;//time it takes for a miss

    switch (argc){//take in arguments if available
        case 6:
            associative = atoi (argv[5]);
        case 5:
            miss_time = atoi (argv[4]);
        case 4:
            hit_time = atoi (argv[3]);
        case 3:
            num_of_blocks = atoi (argv[2]);
        case 2:
            block_size = atoi(argv[1]);
    }
```

You can see the default values set.
After it reads parameters, it checks to make sure associativity is one and creates the cache.

```
int block_array[num_of_blocks][block_size];//create cache
if(associative != 1){//Direct map only
    cout << "Please make sure associative is 1 (The 5th variable on command line)" << endl;
    return 0;
}
```

The address file is then opened

```
fstream newfile;
newfile.open("addresses.txt", fstream::in);//open address file
string line;
float hit, total;
total = 0;
hit = 0;
```

We then read every line of the address file, each line being an address. The address is then turned to decimal, and calculates the block remainder and finds the block number. This is then used to find where in the cache the address should be stored. Once we find the location, we check to see if it is already there. If it is, then it's a hit, and we move on to the next line. If it is not, then we need to replace what is currently there with the new address.

```
while(getline(newfile, line)){//each line/address
    total++;
    line.erase(0, 2);
    int value;
    value = stoi(line, 0, 16);
    int block_add, block_num;
    block_add = value/block_size;//find address in block
    int block_rem = value % block_size;
    block_num = block_add % num_of_blocks;//find which block
    int current_address = block_array[block_num][block_rem];
    if(current_address == value){//check to see if block already has address
        hit++;
    }
    else{
        block_array[block_num][block_rem] = value;//if not switch it out
    }
}
```

After the entire text file is read, we continue on to calculating the hit miss rate and the AMAT. The hit miss rate is easy, as it is simply the hit count over the total, miss rate over the total, and hit rate percent. The AMAT is calculated using its formula of hit time + miss rate * miss penalty. All of this is printed out for the user.

```
float hitmisspercent = (hit/total) * 100;//cal hitmisspercent
float miss = total - hit;
float missrate = miss/total;//miss rate
cout << "Hit rate: " << hit << "/" << total << endl;
cout << "Miss rate: " << miss << "/" << total << endl;
cout << "Hit miss rate: " << hitmisspercent << "%" << endl;
float AMAT = hit_time + missrate * miss_time;
cout << "AMAT: " << AMAT << " ns" << endl;
```

I tested this code by changing the addresses.txt and using smaller values. I also checked the larger address txt by finding the number of repeats and increasing my parameters to have enough room for every address.