The University of Texas at Dallas

ASIC Assignment-3

AXM220237, UXK220003

Adithya Prathipna Mullan Koni, Usman Khan
2-10-2024

C++ Code Implementation

This code is designed to perform a convolution operation using hexadecimal values stored in files. Here's a brief explanation of how it works:

1. **File Input**: The program takes three command-line arguments: filename, filenamecoeff, and filenameout. These filenames correspond to the input data, coefficients, and output file, respectively.
2. **Reading Hexadecimal Values**: The readHexadecimalValuesFromFile function reads hexadecimal values from the input file specified by filename and stores them in a vector of unsigned integers.
3. **Formatting Hexadecimal Values**: The readAndFormatHexValues function reads hexadecimal values from the input file specified by filename, formats them according to certain conditions (e.g., prepend "FFFFFF" if the most significant bit is set), and stores them in a vector of long integers.
4. **Two's Complement Function**: The twos_comp function computes the two's complement of a given number.
5. **Multiplication Function**: The multiplication function performs multiplication of two numbers in the context of the convolution operation.
6. **Convolution Function**: The convolution function implements the convolution operation using the input data array and coefficient array.
7. **Main Function**: The main function orchestrates the entire process. It reads the input data and coefficients, performs the convolution operation, formats the result, and writes it to the output file specified by filenameout.
8. **File Output**: The result of the convolution operation is written to the output file specified by filenameout, with each result value written on a separate line.

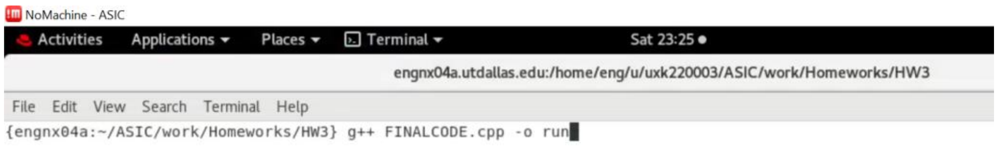
This code provides a flexible and efficient way to perform convolution operations on hexadecimal data stored in files.

Code

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <iomanip>
#include <cstdlib>
using namespace std;

std::vector<unsigned int> readHexadecimalValuesFromFile(const std::string& filename) {
    std::vector<unsigned int> hexValues;
    std::ifstream inputFile(filename.c_str());

    if (!inputFile.is_open()) {
        std::cerr << "Error opening the file: " << filename << std::endl;
        return hexValues; // Return empty vector
    }

    std::string hexValue;
    while (inputFile >> hexValue) {
        // Convert hexadecimal string to integer and store it in the vector
        unsigned int decimalValue;
        std::stringstream ss;
        ss << std::hex << hexValue;
        ss >> decimalValue;
        hexValues.push_back(decimalValue);
    }

    inputFile.close();
    return hexValues;
}

std::vector<long int> readAndFormatHexValues(const std::string& filename) {
    std::vector<long int> formattedHexValuesAsLong;
    std::ifstream file(filename.c_str());
    if (!file.is_open()) {
        std::cerr << "Error opening file: " << filename << std::endl;
        return formattedHexValuesAsLong;
    }

    std::string line;
    while (std::getline(file, line)) {
        std::istringstream iss(line);
        std::string hexValue;
        while (iss >> hexValue) {
            long int x;
            std::stringstream ss;
            ss << std::hex << hexValue;
            ss >> x;

            std::ostringstream formattedStream;
            if ((x & 0x8000) == 0x8000) {
                formattedStream << "FFFFFF" << std::uppercase << std::setfill('0') << std::setw(4) << std::hex <<
x << "0000";
            } else {
                formattedStream << "000000" << std::uppercase << std::setfill('0') << std::setw(4) << std::hex
<< x << "0000";
            }

            formattedHexValuesAsLong.push_back(strtol(formattedStream.str().c_str(), NULL, 16));
        }
    }

    file.close();
    return formattedHexValuesAsLong;
}

long int twos_comp(long int num){
    num = ~num;
    return num +1;
}

long int multiplication(long int x, unsigned int h){
    long int input = 0, temp,temp1;
    long int result = 0;
    int i;
    for(i =0;i<16;i++){
        if((h>>i)&1!=0)
            break;
    }
    while(i<16){
        input = x;
        temp =0;
        temp1 = h>>i;
        if((temp1&1)!=0)
        {
            temp1 = temp1>>16;
            if((temp1&1) == 1)
                temp = twos_comp(input);
            else
                temp = input;
        }
        result+= temp;
        result = result>>1;
        i++;
    }
    return result;
}

long int* convolution(long int x_arr[], unsigned int h_arr[], int size){
    int n,k;
    long int* y = new long int[size];
    long int temp;
    for(n=0;n<size;n++)
    {
        temp =0;
        for(k=0;k<256;k++){
            if((n-k)<0)
                break;
            temp += multiplication(x_arr[n-k],h_arr[k]);
        }
        y[n] = temp;
    }
    return y;
}

int main(int argc, char *argv[])
{
    if (argc != 4) {
        std::cerr << "Usage: " << argv[0] << " filename filenamecoeff filenameout" << std::endl;
        return 1;
    }

    std::string filename = argv[1];
    std::string filenamecoeff = argv[2];
    std::string filenameout = argv[3];

    long int* result_int;
    std::string result[1000];

    std::vector<long int> formattedHexValuesAsLong = readAndFormatHexValues(filename);
    long int xyz[1000];

    for (size_t i = 0; i < formattedHexValuesAsLong.size(); ++i)
    {
        xyz[i] = formattedHexValuesAsLong[i];
    }

    std::vector<unsigned int> hexValues = readHexadecimalValuesFromFile(filenamecoeff);
    unsigned int coeff[1000];

    for (size_t j = 0; j < hexValues.size(); ++j)
    {
        coeff[j] = hexValues[j];
    }

    int size = sizeof(xyz)/sizeof(xyz[0]);
    result_int = convolution(xyz,coeff,size);
    int len;


    for(int i=0;i<size;i++)
    {
        string zeros ("");
        std::stringstream ss;
        ss<<uppercase<<std::hex<<result_int[i];

        result[i] = (ss.str());
        len = 16 - result[i].size();
        while(len > 0 )
        {
            zeros = zeros + "0";
            len--;
        }
        result[i]  = (zeros + result[i]).substr(6);

        // std::cout << result[i] << std::endl;
    }

    std::ofstream outputFile(filenameout.c_str());
    if (!outputFile.is_open()) {
        std::cerr << "Error opening the file: " << filenameout << std::endl;
        return 1;
    }

    for (int i = 0; i < size; i++) {
        outputFile << result[i] << std::endl;
    }
    outputFile.close();

    return 0;
}
```

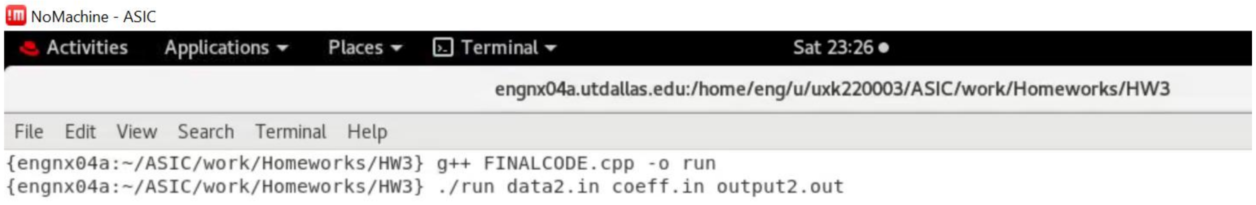
How to Compile and Run:

Run the command in the command line and press enter


NoMachine - ASIC
Activities
Applications
Places
Terminal
Sat 23:25
engnx04a.utdallas.edu:/home/eng/u/uxk220003/ASIC/work/Homeworks/HW3
File Edit View Search Terminal Help

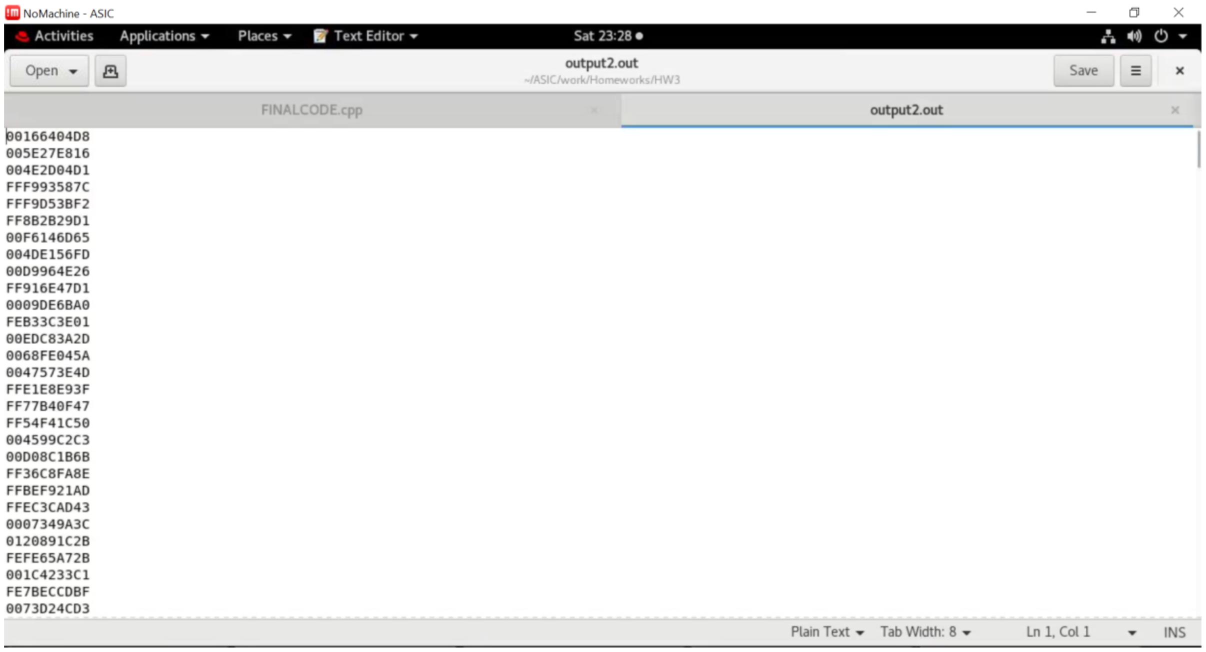
```
{engnx04a:~/ASIC/work/Homeworks/HW3} g++ FINALCODE.cpp -o run
```

Next, run this command in the command line and press enter.


NoMachine - ASIC
Activities
Applications
Places
Terminal
Sat 23:26
engnx04a.utdallas.edu:/home/eng/u/uxk220003/ASIC/work/Homeworks/HW3
File Edit View Search Terminal Help

```
{engnx04a:~/ASIC/work/Homeworks/HW3} g++ FINALCODE.cpp -o run
{engnx04a:~/ASIC/work/Homeworks/HW3} ./run data2.in coeff.in output2.out
```

Now go to output2.out file and check for the outputted value.



```
00166404D8
005E27E816
004E2D04D1
FFF993587C
FFF9D53BF2
FF8B2B29D1
00F6146D65
004DE156FD
00D9964E26
FF916E47D1
0009DE6BA0
FEB33C3E01
00EDC83A2D
0068FE045A
0047573E4D
FFE1E8E93F
FF77B40F47
FF54F41C50
004599C2C3
00D08C1B6B
FF36C8FA8E
FFBEF921AD
FFEC3CAD43
0007349A3C
0120891C2B
FEFE65A72B
001C4233C1
FE7BECCDBF
0073D24CD3
```