Code for macro pass2

```
#include <bits/stdc++.h>
using namespace std;

// Structure for the Macro Name Table (MNT)
struct MNTEntry {
  string macroName;
  int mdtIndex;
};

// Global variables for MNT, MDT, and ALA
unordered_map<string, MNTEntry> MNT;
vector<string> MDT;
unordered_map<string, string> ALA;

// Function to split a line into tokens
vector<string> splitLine(const string& line) {
  vector<string> tokens;
  stringstream ss(line);
  string word;
  while (ss >> word) {
    tokens.push_back(word);
  }
  return tokens;
}

// Function to load MNT from file
void loadMNT(const string& fileName) {
  ifstream file(fileName);
  if (!file.is_open()) {
    cerr << "Error opening MNT file!" << endl;
    return;
  }

  string macroName;
  int mdtIndex;
  while (file >> macroName >> mdtIndex) {
    MNT[macroName] = {macroName, mdtIndex};
  }

  file.close();
}

// Function to load MDT from file
void loadMDT(const string& fileName) {
  ifstream file(fileName);
  if (!file.is_open()) {
    cerr << "Error opening MDT file!" << endl;
    return;
  }

  string line;
  while (getline(file, line)) {
    MDT.push_back(line);
  }
```

```
    file.close();
}

// Function to load ALA from file
void loadALA(const string& fileName) {
    ifstream file(fileName);
    if (!file.is_open()) {
        cerr << "Error opening ALA file!" << endl;
        return;
    }

    int index;
    string argument;
    while (file >> index >> argument) {
        ALA["&ARG" + to_string(index + 1)] = argument;  // Assuming &ARG1, &ARG2, etc.
    }

    file.close();
}

// Function to expand a macro using MDT and MNT
void expandMacro(const string& macroName, ofstream &outputFile) {
    if (MNT.find(macroName) == MNT.end()) {
        cerr << "Error: Macro " << macroName << " not found!" << endl;
        return;
    }

    int mdtIndex = MNT[macroName].mdtIndex;

    // Process the MDT lines for the given macro
    while (MDT[mdtIndex] != "MEND") {
        string line = MDT[mdtIndex];
        vector<string> tokens = splitLine(line);

        // Replace arguments with actual values from ALA
        for (string& token : tokens) {
            if (ALA.find(token) != ALA.end()) {
                token = ALA[token];  // Replace argument placeholder with actual value
            }
        }

        // Write the expanded macro line to output file
        for (const string& token : tokens) {
            outputFile << token << " ";
        }
        outputFile << endl;

        ++mdtIndex;  // Move to the next MDT entry
    }
}

// Function to process the input assembly file and expand macros
void macroPass2(const string& inputFileName, const string& outputFileName) {
    ifstream inputFile(inputFileName);
    ofstream outputFile(outputFileName);

    if (!inputFile.is_open()) {
```

```
        cerr << "Error opening input assembly file!" << endl;
        return;
    }
    if (!outputFile.is_open()) {
        cerr << "Error opening output file!" << endl;
        return;
    }

    string line;
    vector<string> tokens;

    // Read input file line by line
    while (getline(inputFile, line)) {
        tokens = splitLine(line);
        if (tokens.empty()) continue;

        // Check if the line contains a macro invocation
        if (MNT.find(tokens[0]) != MNT.end()) {
            // It's a macro invocation, expand the macro
            for (size_t i = 1; i < tokens.size(); ++i) {
                ALA["&ARG" + to_string(i)] = tokens[i];  // Store actual arguments in ALA
            }
            expandMacro(tokens[0], outputFile);
        } else {
            // Not a macro, write the line directly to the output file
            outputFile << line << endl;
        }
    }

    inputFile.close();
    outputFile.close();
}

int main() {
    string inputFileName, outputFileName;

    // Load MNT, MDT, and ALA files generated from Macro Pass 1
    loadMNT("MNT.txt");
    loadMDT("MDT.txt");
    loadALA("ALA.txt");

    // Prompt for input assembly file and output file
    cout << "Enter the input assembly file for Pass 2 (e.g., input.asm): ";
    cin >> inputFileName;
    cout << "Enter the output file name for expanded code (e.g., output.asm): ";
    cin >> outputFileName;

    // Perform Macro Pass 2 (expand macros)
    macroPass2(inputFileName, outputFileName);

    cout << "Macro Pass 2 completed. Expanded code written to " << outputFileName << endl;

    return 0;
}
```