Om Shete
TE C32
2/03/23

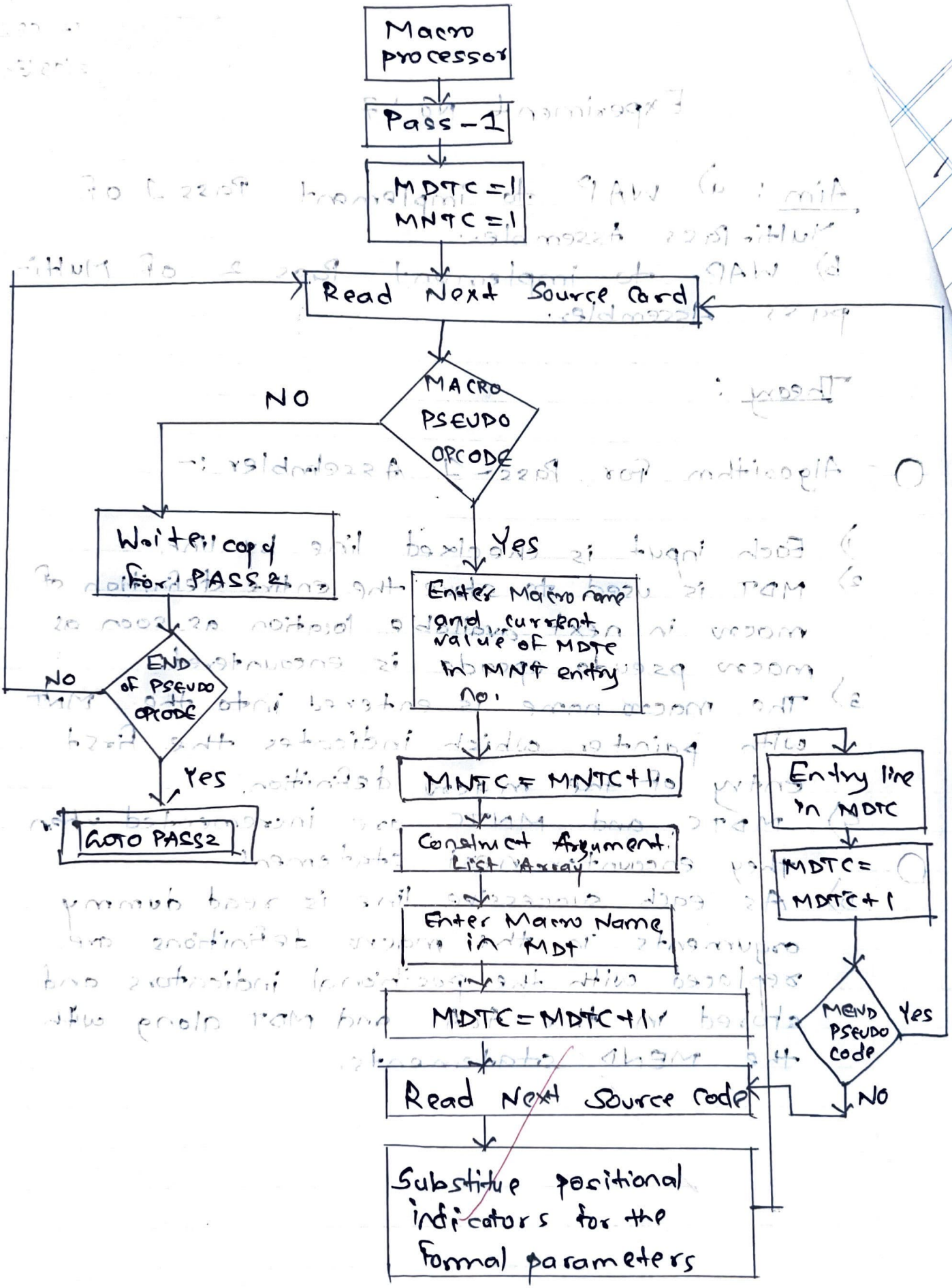
Experiment No : 7

Aim : a) WAP to implement Pass 1 of Multi-Pass Assembler.
b) WAP to implement Pass 2 of Multi-pass Assembler.

Theory :

- Algorithm for Pass-1 Assembler :-

1) Each input is checked line by line.
2) MDT is used to store the entire definition of macro in next available location as soon as macro pseudo opcode is encountered.
3) The macro name is entered into the MNT with pointer which indicates the first entry of the macro definition.
4) MDTC and MNTC are incremented when they encounter next statement.
5) As each successive line is read dummy arguments in the macro definitions are replaced with the positional indicators and stored in the ALA and MDT along with the MEND statements.

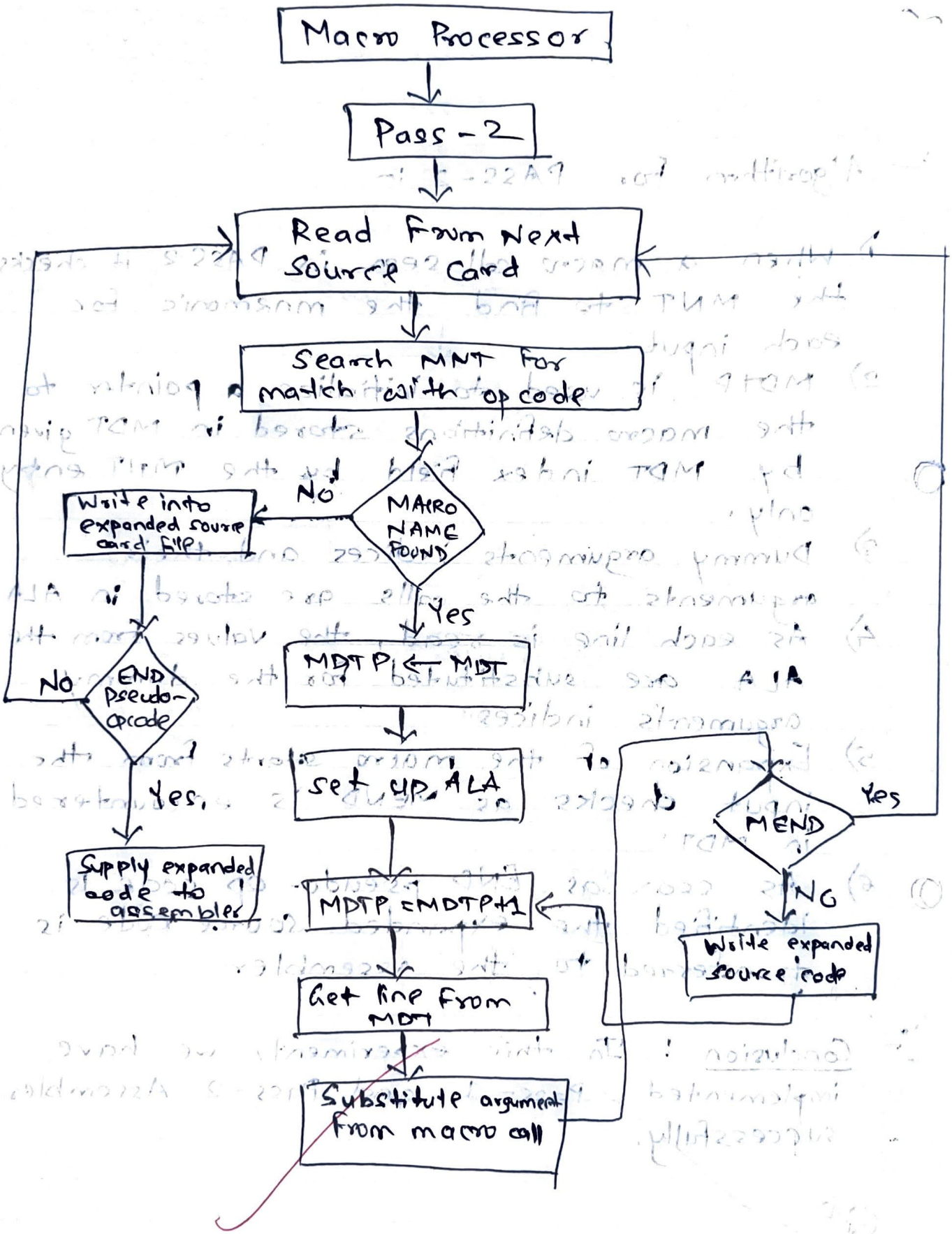

Macro processor
Pass-1
MDTC=1
MNTC=1
Read Next Source Card
MACRO PSEUDO OPCODE
NO
Yes
Write copy for PASS2
Enter Macro name and current value of MDTC in MNT entry no.
END of PSEUDO OPCODE
NO
Yes
GOTO PASS2
MNTC = MNTC+1
Construct Argument List Array
Enter Macro Name in MDT
MDTC=MDTC+1
Read Next Source code
Substitute positional indicators for the formal parameters
Entry line in MDTC
MDTC= MDTC+1
MEND PSEUDO code
Yes
NO

- Algorithm for PASS-2 :-

1) When a macro call seen in PASS 2 it checks the MNT to find the mnemonic for each input.
2) MDTP is used to initialize a pointer to the macro definition's stored in MDT given by MDT index field by the MNT entry only.
3) Dummy arguments indices and their arguments to the calls are stored in ALA
4) As each line is read, the values from the ALA are substituted for the dummy arguments indices.
5) Expansion of the macro starts from the input checks as MEND is encountered in MDT.
6) As soon as END pseudo-op code is identified the expanded source code is transferred to the assembler.

- Conclusion : In this experiment, we have implemented Pass-1 and Pass-2 Assembler successfully.

10/04/2024.


Macro Processor
Pass - 2
Read From Next Source Card
Search MNT for match with op code
MACRO NAME FOUND
No
Write into expanded source card file
END pseudo-opcode
NO
Yes
Supply expanded code to assembler
Yes
MDTP ← MDT
set UP ALA
MDTP = MDTP+1
Get line From MDT
Substitute argument From macro call
MEND
Yes
NO
Write expanded source code

Code:

```
#include <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
#include <vector>

using namespace std;

map<string, pair<int, vector<string>>> MNT; // Macro Name Table
map<int, pair<string, vector<string>>> MXT; // Macro Expansion Table
map<string, string> ALA;                    // Argument List Array

void passOne(ifstream &inputFile);
void passTwo(ifstream &inputFile, ofstream &outputFile);

int main() {
  ifstream inputFile("input.txt");

  if (!inputFile) {
    cerr << "Error: Could not open input file." << endl;
    return 1;
  }

  passOne(inputFile);
  inputFile.close();

  cout << "MNT after pass one:" << endl;
  for (const auto &entry : MNT) {
    cout << entry.first << " -> ";
    cout << "Index: " << entry.second.first << ", ";
    cout << "Args: ";
    for (const auto &arg : entry.second.second) {
      cout << arg << " ";
    }
    cout << endl;
  }
  cout << endl;

  ofstream outputFile("output.txt");
  if (!outputFile) {
```

```
    cerr << "Error: Could not create output file." << endl;
    return 1;
  }

  inputFile.open("input.txt");
  passTwo(inputFile, outputFile); // Second pass
  inputFile.close();
  outputFile.close();

  cout << "MXT after pass two:" << endl;
  for (const auto &entry : MXT) {
    cout << entry.first << " -> ";
    cout << "Macro: " << entry.second.first << ", ";
    cout << "Args: ";
    for (const auto &arg : entry.second.second) {
      cout << arg << " ";
    }
    cout << endl;
  }
  cout << endl;

  cout << "ALA after pass two:" << endl;
  for (const auto &entry : ALA) {
    cout << entry.first << " -> " << entry.second << endl;
  }

  cout << "Assembly process complete. Output written to output.txt" <<
endl;

  return 0;
}

void passOne(ifstream &inputFile) {
  string line;
  int macroIndex = 0;

  while (getline(inputFile, line)) {
    stringstream ss(line);
    string token;
    ss >> token;

    if (token == "MEND") {
      continue;
```

```
    } else if (token == "&LAB") {
      string macroName;
      ss >> macroName;

      MNT[macroName].first = macroIndex++;

      vector<string> arguments;
      while (ss >> token) {
        if (token != ",")
          arguments.push_back(token);
      }

      MNT[macroName].second = arguments;
    }
  }

  cout << "MNT after pass one:" << endl;
  for (const auto &entry : MNT) {
    cout << entry.first << " -> ";
    cout << "Index: " << entry.second.first << ", ";
    cout << "Args: ";
    for (const auto &arg : entry.second.second) {
      cout << arg << " ";
    }
    cout << endl;
  }
  cout << endl;
}

void passTwo(ifstream &inputFile, ofstream &outputFile) {
  string line;
  int macroIndex = 0;

  while (getline(inputFile, line)) {
    stringstream ss(line);
    string token;

    while (ss >> token) {
      if (token == "MEND") {
        continue;
      } else if (token == "&LAB") {
        string macroName;
        ss >> macroName;
```

```
    vector<string> arguments;
    while (ss >> token) {
      if (token != ",")
        arguments.push_back(token);
    }

    MXT[macroIndex].first = macroName;
    MXT[macroIndex].second = arguments;

    macroIndex++;
  } else {

    if (MNT.find(token) != MNT.end()) {

      outputFile << "loop " << macroIndex + 1 << " "
                 << MXT[macroIndex].first << endl;

      int argNum = 1;
      for (const auto &arg : MNT[token].second) {
        if (arg[0] == '%') {

          string argValue = ALA[arg];
          outputFile << "  " << argNum << " " << argValue << " ";
        } else {
          outputFile << "  " << argNum << " " << arg << " ";
        }

        ss >> token;
        outputFile << " " << argNum << ", " << token << endl;
        argNum++;
      }
    } else {

      string macroName = MXT[macroIndex].first;
      vector<string> args = MXT[macroIndex].second;
      for (size_t i = 0; i < args.size(); ++i) {
        ALA[args[i]] = token;

        ss >> token;
      }
    }
  }
```

```
    }
  }

  cout << "MXT after pass two:" << endl;
  for (const auto &entry : MXT) {
    cout << entry.first << " -> ";
    cout << "Macro: " << entry.second.first << ", ";
    cout << "Args: ";
    for (const auto &arg : entry.second.second) {
      cout << arg << " ";
    }
    cout << endl;
  }
  cout << endl;

  // cout << "ALA after pass two:" << endl;
  // for (const auto& entry : ALA) {
  //     cout << entry.first << " -> " << entry.second << endl;
  // }
}
```

main.cpp × input.txt × output.txt +

input.txt

```
&LAB INCR &ARG1, &ARG2, &ARG3
&LAB A 1, %AR1
&LAB B 1, %AR1
&LAB C 1, %AR1
MEND

Loop1 INCR data3, data2, data1
Loop2 INCR data1, data2, data3
```

Output:

```
Run                                                    Ask AI   6s on 20:55:28, 04/04 ✓

MNT after pass one:
A -> Index: 1, Args: 1, %AR1
B -> Index: 2, Args: 1, %AR1
C -> Index: 3, Args: 1, %AR1
INCR -> Index: 0, Args: &ARG1, &ARG2, &ARG3

MNT after pass one:
A -> Index: 1, Args: 1, %AR1
B -> Index: 2, Args: 1, %AR1
C -> Index: 3, Args: 1, %AR1
INCR -> Index: 0, Args: &ARG1, &ARG2, &ARG3

MXT after pass two:
0 -> Macro: INCR, Args: &ARG1, &ARG2, &ARG3
1 -> Macro: A, Args: 1, %AR1
2 -> Macro: B, Args: 1, %AR1
3 -> Macro: C, Args: 1, %AR1
4 -> Macro: , Args:

MXT after pass two:
0 -> Macro: INCR, Args: &ARG1, &ARG2, &ARG3
1 -> Macro: A, Args: 1, %AR1
2 -> Macro: B, Args: 1, %AR1
3 -> Macro: C, Args: 1, %AR1
4 -> Macro: , Args:

ALA after pass two:
Assembly process complete. Output written to output.txt
```

main.cpp × input.txt output.txt × +

output.txt

```
loop 5
  1 &ARG1,  1, data3,
  2 &ARG2,  2, data2,
  3 &ARG3  3, data1
loop 5
  1 &ARG1,  1, data1,
  2 &ARG2,  2, data2,
  3 &ARG3  3, data3

```