Date

Assignment - A2

Title:- Pass-II of Two pass assembler

Problem Statement:-

Implement pass-II of two pass assembler for pseudo-machine in java using object oriented features. The output of assignment-1 should be input i/p of this assignment.

objective :-

① Understand the internals of language translators

② Handle tools like LEX & YACC

③ Understand the operating System internals & functionalities with the implementation point of view.

H/W & S/W :-

System with 64bit OS, Eclipse, Java 13 & 25 machines.

Theory:-

Assembler is a program which convers assembly language instructions into machine language form. A two pass assembler takes two scans of sorce code to produce the machine code to produce the machine code from assembly language program.

Assembly process consists of following activity :

1. Convert mnemonics to their machine language opcode equivalent
2. Convert symbolic operands to their machine address
3. Translate data constants into internal machine representation.
4. Output the object program & provide other information required for linker & loader.

Pass-II tasks :-

1. Generate opcode data values defined by BYTE, WORD.
2. Assembler Instructions (generate opcode & lookup address).

③ Perform processing of assembler directives (not done in pass-I)

④ Write the object program & the assembly listing.

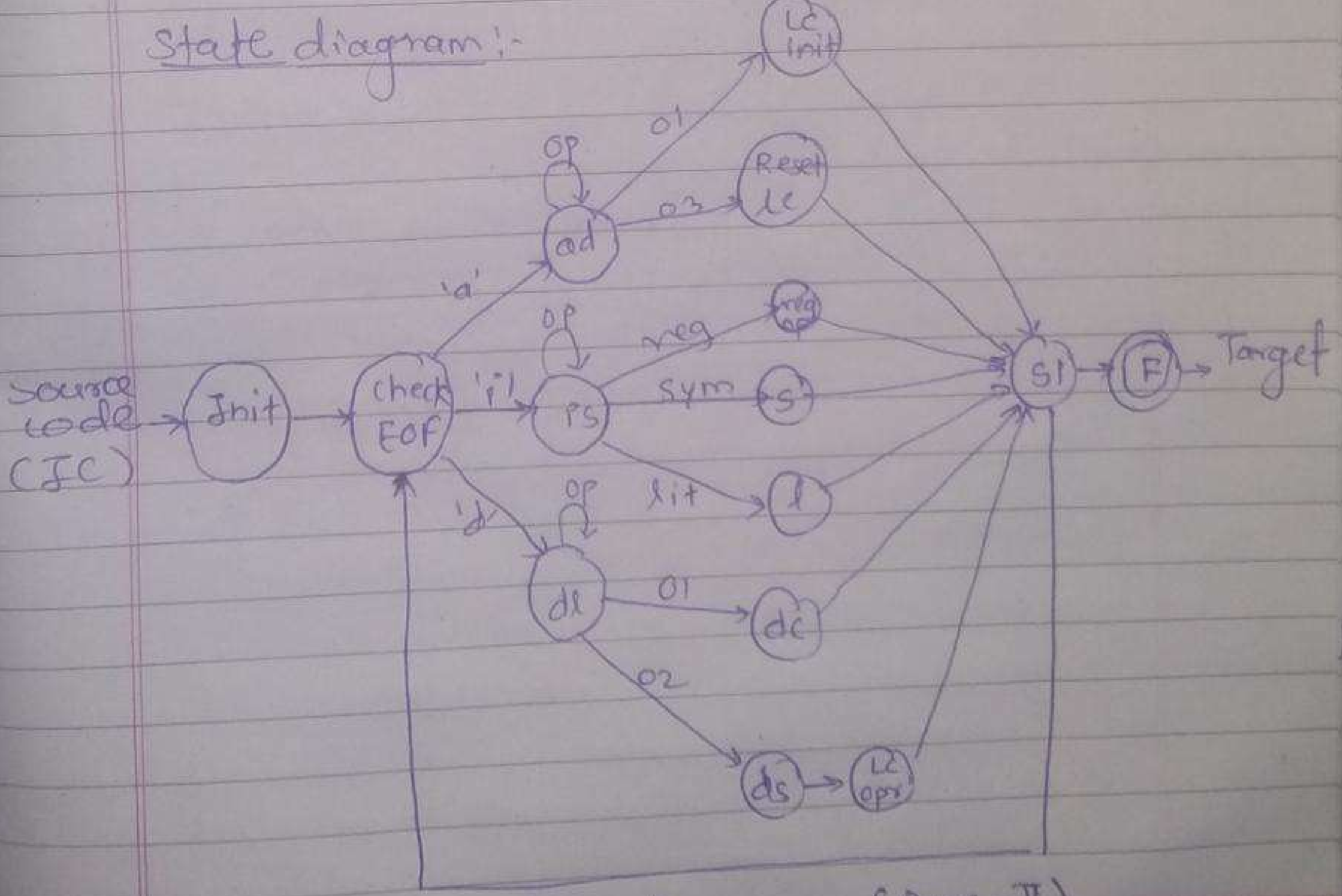
Algorithm :-

① Read intermediate code file generated in Pass-I

② Search symbol & literal tables to use in machine code generation.

③ Generate machine code

State diagram :-



State diagram (Pass-II)

Conclusion:-

We have learnt & successfully implemented the pass-II assembler.

```
package asm;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

public class pass2 {

 public static void main(String[] args) throws IOException {
  String line;
  String[][] mc_code= new String[10][3] ;
  int count=0;
   BufferedReader b1 = new BufferedReader(new FileReader("IC.txt"));
   BufferedReader b2 = new BufferedReader(new FileReader("symtab.txt"));
   BufferedReader b3 = new BufferedReader(new FileReader("littab.txt"));
   HashMap <Integer,String>symaddr=new HashMap <Integer,String>();
   HashMap <Integer,String>litaddr =new HashMap <Integer,String>();



    System.out.println();
    System.out.println("\n\t ======= SYMBOL TABLE ========");
     System.out.println("\t----------------------------------");
     System.out.println("\tSymbol|"+"\t"+"Address");
     System.out.println("\t----------------------------------");
  while((line = b2.readLine()) != null)
     {
    String split_words[] = line.split("\t");
    count++;
    System.out.println("\t" + line);
    symaddr.put(count, split_words[1]);
     }
  System.out.println("\n");

  System.out.println("\n\t ======= Literal TABLE =======");
     System.out.println("\t--------------------------------");
     System.out.println("\tLiteral|"+"Address");
     System.out.println("\t--------------------------------");
  count=0;
  while((line = b3.readLine()) != null)
     {
    String split_words[] = line.split("\t");
    System.out.println("\t "+line);
    count++;
    litaddr.put(count, split_words[1]);
     }

  System.out.println("\n");

  System.out.println("\n\t ======= OPCODE TABLE ======= \n");
```

```
   System.out.println("\t----------------------------------------");
   System.out.println("\tMnemonic|"+"Info");
   System.out.println("\t---------------------------------------");
   count=0;
while((line = b1.readLine()) != null)
{
 String split_words[] = line.split("\t");
 System.out.println("\t" + line);
 if(split_words[1].contains("IS"))
 {
  mc_code[count][0]=split_words[1].substring(4,5);
  mc_code[count][1]=split_words[2];
  if(split_words[3].contains("C"))
  {
   int lit_index=Integer.parseInt(split_words[3].substring(3,4));
   mc_code[count][2]=litaddr.get(lit_index);
  }
  else if(split_words[3].contains("S"))
  {
   int sym_index=Integer.parseInt(split_words[3].substring(3,4));
   mc_code[count][2]=symaddr.get(sym_index);
  }
  count++;
 }
 else if(split_words[1].contains("DL,1"))
 {
  mc_code[count][0]=  "00" ;
  mc_code[count][1]= "0" ;
  mc_code[count][2]= "00"+ split_words[2].substring(3,4);
  count++;
 }

}
System.out.println("\n");
System.out.println("\n\t================ MACHINE CODE TABLE ================");
   System.out.println("\t-----------------------------------------------");
System.out.println("\tIS op-code\t"+"|Register\t"+"|Symbol address");
System.out.println("\t---------------------------------------------");
for(int i=0;i<count;i++)
{
 System.out.println("\t"+mc_code[i][0]+"\t\t"+mc_code[i][1]+"\t\t"+mc_code[i][2]);
}

File mcode = new File("MC.txt");
FileWriter wr = new FileWriter("MC.txt");


 for(int i=0;i<count;i++)
 {
  wr.write(mc_code[i][0]+"\t"+mc_code[i][1]+"\t"+mc_code[i][2]+"\n");
 }
 wr.close();
 }

}
```

```
 ======= SYMBOL TABLE ========
-----------------------------------
Symbol| Address
-----------------------------------
A 200
B 202



 ======= Literal TABLE =======
---------------------------------
Literal|Address
---------------------------------
 =10 203



 ======= OPCODE TABLE =======

-----------------------------------------
Mnemonic|Info
----------------------------------------
START (AD,1)
DS (DL,2) (C,2)
ADD (IS,1) (31) (C,1)
LTORG (AD,4)
MUL (IS,3) (32) (S,1)
END (AD,2)



============== MACHINE CODE TABLE ==============
------------------------------------------------
IS op-code |Register |Symbol address
------------------------------------------------
1  (31)  203
3  (32)  200
```