Fokane Sakshi Anil
TE-A-42

Assignment No:- 2

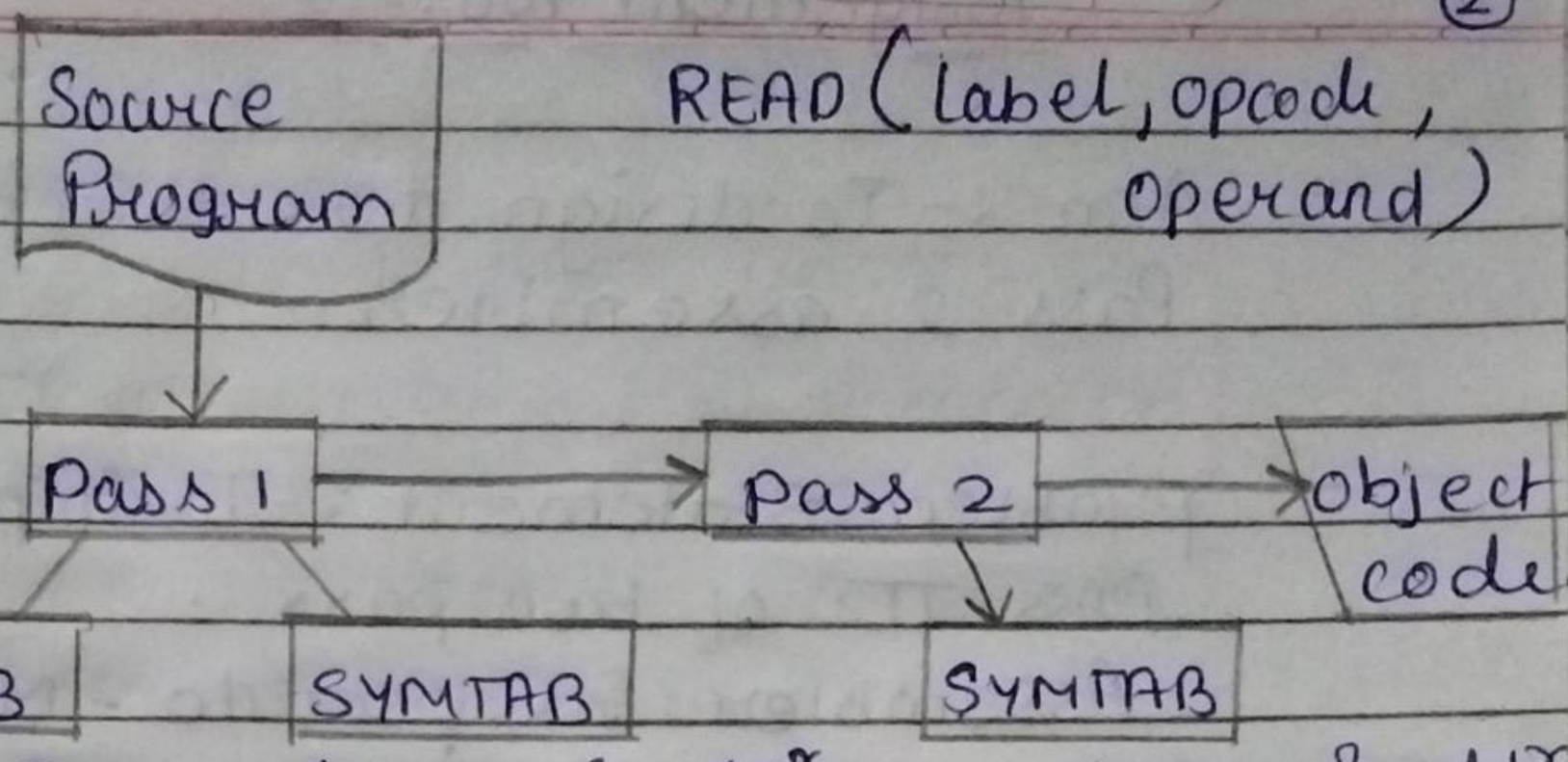
- Aim :- To design data structure for Pass 2 assembler

- problem Statement :- To implement PASS-II of two pass Assembler for pseudo-Machine in java using object oriented features. The output of assignment-1 (Intermediate file and Symbol table) should be input for this assignment.

- Theory :-

Two pass assembler :

The two pass assembler performs two passes over the source program. In the first pass, it reads the entires source programs, looking only for label. defination all the labels are collected assigned address, & placed in the symbol table. in the programs. No instruction as assembled & at the end the Symbol table should contain all the labels defines in the programs to design address to labels. the assembles maintain a location counter



* Difference between onepass & two pass Assembler.

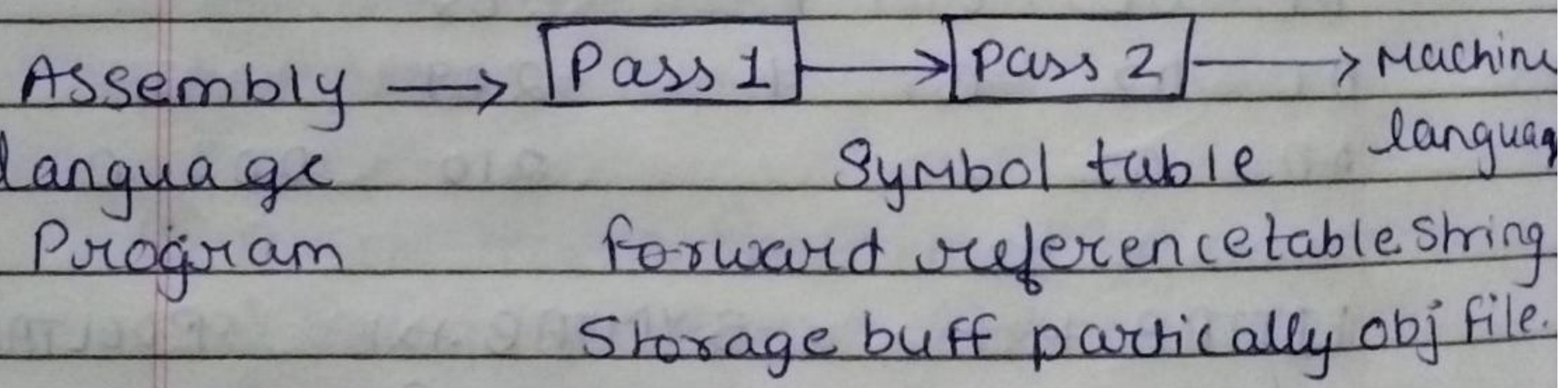
A one pass assembler passes over the source file exactly once, in the same pass collecting the tables, resolving future references & doing the actual assembly. The difficult part is to resolve future label refere-nce & assemble code in one pass. The one pass assembler prepare an intermediate file which is used as input by the two pass assembler.

The two pass assembler performs two sequential scan over the source code.

pass 1 : Symbol & literals are defined
pass 2 : object program is generated

parsing : moving in program lines to pull out op-code & and operands

* Data structure :
- location counter (LC) : point to the next location where the code will be placed
- op-code translation table : contains symbolic instruction their lengths & their op-codes.
- Symbol table (ST) : Contain labels & their values.
- String Storage buffer (SSB) : Contain ASCII characters for the Strings.



* Algorithm :-

```
begin
    if starting add^r is given
        LOCCTR = starting add^r
    else
        LOCCTR = 0;
    while OPCODE != END do    ::or EOF
begin
    read a line from the code
    if there is a label
    if this label is in SYMTAB then error
    else insert (label, LOCCTR) into SYMTAB
    search OPTAB for the op-code
if found
        LOCCTR += N
else if this is an assembly directive
update LOCCTR as directed.
```

else error
        write line to intermediate file
end
end. program size = LOCCTR - start addr;

Input

IC.txt

AD	01	C	200					
IS	04	1	1	1				
IS	05	1	S	1				
IS	04	2	1	2				
IS	04	3	S	3				
AD	05							
IS	01	3	L	3				
IS	00							
DL	02	C	1					
DL	02	C	1					
AD	02							

Expected O/P

				200	04	1	204	
				201	05	1	208	
				202	04	2	210	
				203	04	3	209	
				204	00	0	004	
				205	00	0	006	
				206	00	3	205	
				207	00	0	000	
				208				
				209				
				210	00	0	001	

LITTAB.txt

='4'	204
='6'	210
='1'	205

SYMTAB.txt

A	208
LOOP	205
B	209

POOLTAB.txt

1
3

Conclusion :-
Thus, we have generated Machine code for the Source program.

```
//Name:Fokane Sakshi Anil
// TE-A 42
//  ASSINGNMENT:GROUP_A_2
/*
Problem Statement: Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented
features. The output of assignment-1 (intermediate file and symbol table) should be
input for this assignment.
*/
import java.io.BufferedReader;
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{
		BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));
	    BufferedReader b2 = new BufferedReader(new FileReader("symtab.txt"));
	    BufferedReader b3 = new BufferedReader(new FileReader("littab.txt"));
	    FileWriter f1 = new FileWriter("Pass2.txt");
	    HashMap<Integer, String> symSymbol = new HashMap<Integer, String>();
	    HashMap<Integer, String> litSymbol = new HashMap<Integer, String>();
	    HashMap<Integer, String> litAddr = new HashMap<Integer, String>();
	    String s;
	    int symtabPointer=1,littabPointer=1,offset;
	    while((s=b2.readLine())!=null){
		    String word[]=s.split("\t\t\t");
		    symSymbol.put(symtabPointer++,word[1]);
	    }
	    while((s=b3.readLine())!=null){
		    String word[]=s.split("\t\t");
		    litSymbol.put(littabPointer,word[0]);
		    litAddr.put(littabPointer++,word[1]);
	    }
	    while((s=b1.readLine())!=null){
		    if(s.substring(1,6).compareToIgnoreCase("IS,00")==0){
			    f1.write("+ 00 0 000\n");
		    }
		    else if(s.substring(1,3).compareToIgnoreCase("IS")==0){
			    f1.write("+ "+s.substring(4,6)+" ");
			    if(s.charAt(9)==')'){
```

```
                                        f1.write(s.charAt(8)+" ");
                                        offset=3;
                                }
                                else{
                                        f1.write("0 ");
                                        offset=0;
                                }
                                if(s.charAt(8+offset)=='S')

f1.write(symSymbol.get(Integer.parseInt(s.substring(10+offset,s.length()-1)))+"\n");
                                else
                                        f1.write(litAddr.get(Integer.parseInt(s.substring(10+offset,s.length()-
1)))+"\n");
                        }
                         else if(s.substring(1,6).compareToIgnoreCase("DL,01")==0){
                                String s1=s.substring(10,s.length()-1),s2="";
                                for(int i=0;i<3-s1.length();i++)
                                        s2+="0";
                                s2+=s1;
                                f1.write("+ 00 0 "+s2+"\n");
                        }
                        else{
                                f1.write("\n");
                        }
                }
                f1.close();
                b1.close();
                b2.close();
                b3.close();
        }
}

/*
OUTPUT:

intermediate code -
(AD,01)(C,200)
(IS,04)(1)(L,1)
(IS,05)(1)(S,1)
(IS,04)(1)(S,1)
(IS,04)(3)(S,3)
(IS,01)(3)(L,2)
```

(IS,07)(6)(S,4)
(DL,01)(C,5)
(DL,01)(C,1)
(IS,02)(1)(L,3)
(IS,07)(1)(S,5)
(IS,00)
(AD,03)(S,2)+2
(IS,03)(3)(S,3)
(AD,03)(S,6)+1
(DL,02)(C,1)
(DL,02)(C,1)
(AD,02)
(DL,01)(C,1)

Symbol Table --

A	211	1
LOOP	202	1
B	212	1
NEXT	208	1
BACK	202	1
LAST	210	1

literal table --

5	206
1	207
1	213

machine code --

+ 04 1 206
+ 05 1 211
+ 04 1 211
+ 04 3 212
+ 01 3 207
+ 07 6 208
+ 00 0 005
+ 00 0 001
+ 02 1 213
+ 07 1 202
+ 00 0 000
+ 03 3 212 */