Akhilesh Anil Shingare
72018540J
COMPTEB083G

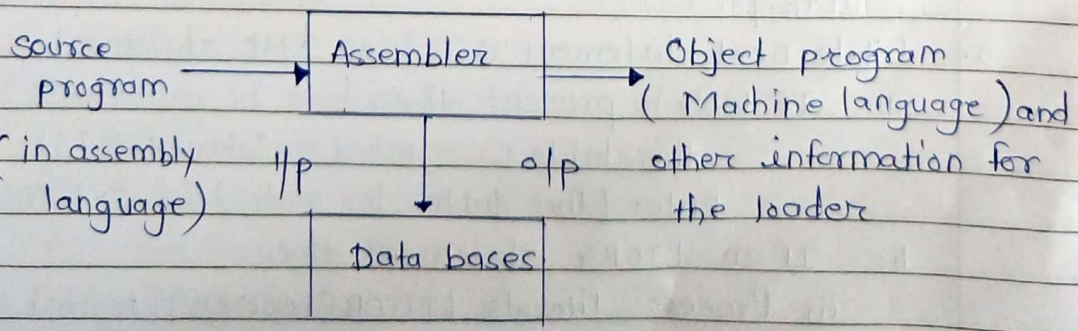
Assignment No. 01

Title:- To design a Pass-1 of two Pass assembler

Problem Statement:- Design suitable data structures and implement pass-1 of two-pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.

Objective:- Design suitable data structures and develope a subset of an assembler subset should consist of a few instructions from each category and few assembler directives

Theory:- An assembler is a computer program for translating assembly language essentially, a mnemonic representation of machine language -into object code. To make writing and reading programs written in machine language more convenient, mnemonic (symbol) were used for each machine instruction. These mnemonics were then later translated into machine language. Such a mnemonic language is called assembly language Assemblers are programs to automatically translate assembly language into machine language.



Other than producing the machine language, the assembler must also produce the information for the loader to use eg externally defined symbols must be noted and passed to the loader.

Data structures:-

Pass I data bases:

1. Input source program
2. A location counter (LC), used to keep track of each instructions location
3. A machine operation table (MOT), that indicates the symbolic mnemonic for each and its length.
4. A pseudo-operation table (POT), that indicates the symbolic mnemonic and action to be taken for each pseudo-op in pass I
5. A symbol table (ST), that is used to store each literal encountered and its corresponding value.
6. A literal table (LT) that is used to store each literal encountered that its corresponding assigned location.
7. A copy of the input, to be passed to pass 2. This is to be stored in storage device.

Algorithm of Pass I:

1. loc-cntr := 0; (default value)
   pooltab-ptr := 1; POOLTAB[1] := 1;
   littab-ptr := 1;
2. (While next statement is not an END statement
   a. If lable is present then
      this-lable := symbol in lable field;
      Enter (this lable, loc cntr) in SYMTAB
   b. If an LTORG statement then
      (i) Process literals LITTAB[POOLTAB][pooltab-ptr]. LITTAB[littab-ptr-1] to allocate memory and put the address in the address field. Update loc-cntr accordingly.

(ii) Pooltab-ptr := pooltab-ptr + 1;
(iii) POOLTAB [pooltab-ptr]: littab-ptr;

c. If a START or ORIGIN statement then
loc-cntr := value specified in operand field

d. IF an EQU statement then
(i) this-addr := value of <address spec>;
(ii) correct the symtab entry for this-lable to (this-lable, this-addr)

e. If a declaration statement then
(i) code := code of the declaration statement;
(ii) size := size of memory area required by DC/DS.
(iii) loc-cntr := loc-cntr + size;
(iv) Generate IC 'C (DL, code) ...'.

f. If an imperative statement then
(i) code := machine opcode from OPTAB;
(ii) loc-cntr := loc-cntr + instruction length from OPTAB;
(iii) If operand is a literal then
this-literal := literal in operand field;
LITTAB [littab-ptr]: = this-literal;
littab-ptr := littab-ptr + 1;
Else (i.e. operand is a symbol)
this-entry := SYMTAB entry number of operand;
Generate IC '(IS, code) (S, this-entry)';

3. (Processing of END statement)
(a) Perform step 2(b)
(b) Generate IC '(AD, 02)'.
(c) Go to Pass II.

Conclusion:- Hence we studied To design a Pass-I of two pass assembler.

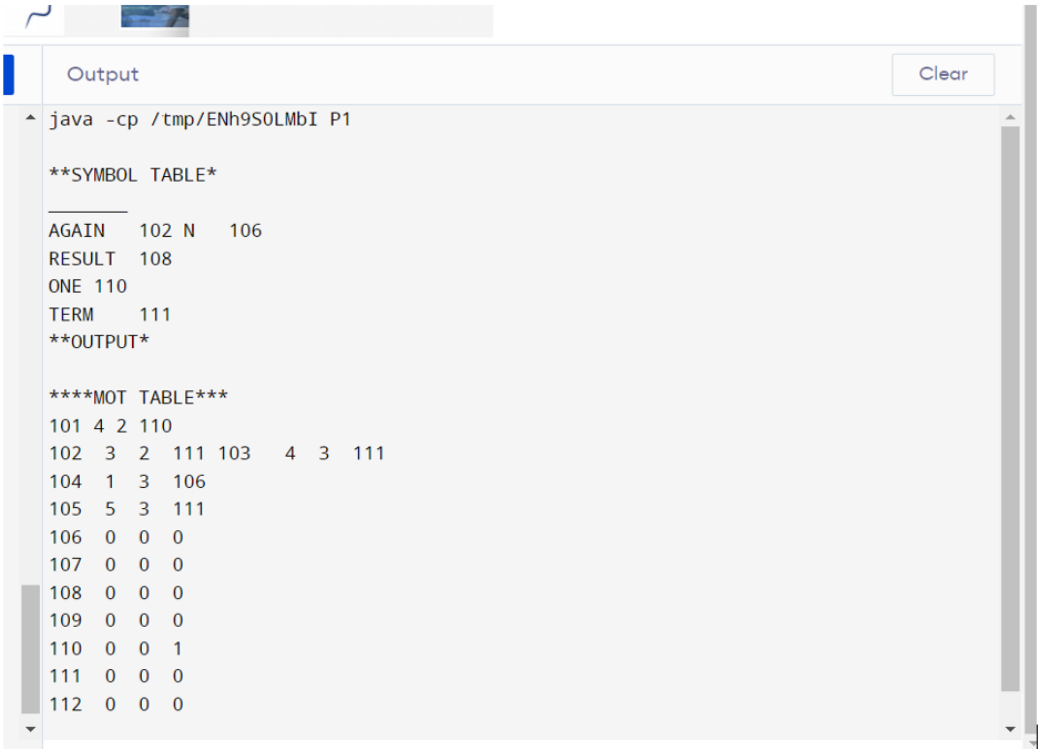
PROGRAM:

```
import java.io.*;
class P1
{
public static void main(String ar[])throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
int i;
String a[][]={{"","START","101",""},
{"","MOVER","BREG","ONE"},
{"AGAIN","MULT","BREG","TERM"},
{"","MOVER","CREG","TERM"},
{"","ADD","CREG","N"},
{"","MOVEM","CREG","TERM"},
{"N","DS","2",""},
{"RESULT","DS","2",""},
{"ONE","DC","1",""},
{"TERM","DS","1",""},
{"","END","",""}};
int lc=Integer.parseInt(a[0][2]);
String st[][]=new String[5][2];
int cnt=0,l;
for (i=1;i<11;i++)
{
if (a[i][0]!="")
{
st [cnt][0]=a[i][0];
st[cnt][1]=Integer.toString(lc);
cnt++;
if(a[i][1]=="DS")
{
int d=Integer.parseInt(a[i][2]);
lc=lc+d;
}
else
{
lc++;
```

```
}
}
else
{
lc++;
}
}
System.out.print("**SYMBOL TABLE*\n"); System.out.println("_______");
for(i=0;i<5;i++)
{
for(cnt=0;cnt<2;cnt++)
{
System.out.print(st[i][cnt]+"\t");
}
System.out.println();
}
String
inst[]={"STOP","ADD","SUB","MULT","MOVER","MOVEM","COMP","BC","DIV","READ
","PRINT"};
String reg[]={"NULL","AREG","BREG","CREG","DREG"};
int op[][]=new int[12][3];
int j,k,p=1,cnt1=0;
for(i=1;i<11;i++)
{
for(j=0;j<11;j++)
{
if(a[i][1].equalsIgnoreCase(inst[j]))
{
op[cnt1][0]=j;
}
else
if(a[i][1].equalsIgnoreCase("DS"))
{
p=Integer.parseInt(a[i][2]);
}
else if(a[i][1].equalsIgnoreCase("DC"))
{
op[cnt1][2]=Integer.parseInt(a[i][2]);
```

```
}
}
for(k=0;k<5;k++)
{
if(a[i][2].equalsIgnoreCase(reg[k]))
{
op[cnt1][1]=k;
}
}
for(l=0;l<5;l++)
{
if(a[i][3].equalsIgnoreCase(st[l][0]))
{
int mn=Integer.parseInt(st[l][1]);
op[cnt1][2]=mn;
}
}
cnt1=cnt1+p;
} System.out.println("\n **OUTPUT*\n"); System.out.println("****MOT TABLE***");
int dlc=Integer.parseInt(a[0][2]);
for(i=0;i<12;i++)
{
System.out.print(dlc+++"\t");
for(j=0;j<3;j++)
{
System.out.print(" "+op[i][j]+" ");
}
System.out.println();
} System.out.println("");
}
}
```

OUTPUT


Output
Clear
java -cp /tmp/ENh9SOLMbI P1
**SYMBOL TABLE*
AGAIN 102 N 106
RESULT 108
ONE 110
TERM 111
**OUTPUT*
****MOT TABLE***
101 4 2 110
102 3 2 111 103 4 3 111
104 1 3 106
105 5 3 111
106 0 0 0
107 0 0 0
108 0 0 0
109 0 0 0
110 0 0 1
111 0 0 0
112 0 0 0