SYSTEM PROGRAMMING LAB WORK-2 (Credits: Sarwat Mir)

**********PROGRAM 1**********

//Simulate working of assembler (PASS 1)FOR GIVEN PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<iostream.h>
#include<string.h>
FILE *f1,*f2,*f3,*f4,*f5,*f6;//f1 is pointer for spp.txt,f2 is for mot, f3 is for symbol table
//f4 is for psuedo op table
int lc,sa,i=0,j=0,k,len1=1; //lc is location counter,len1 keeps track of instr length
char name[20],opnd[30],la[20],mne[20],s1[20];
//these hold various parts of inst

void addsymbol()
{    //if(strcmp(la,"BEGIN")==0)    //for    first instru
       if(strcmp(mne,"BALR")==0) //if mnemonic is balr length of instruction is 2
     len1=2;
   else if(strcmp(mne,"START")==0)
     len1=1;
     else
     len1=4;

   if(strcmp(la,"X")!=0 && strcmp(mne,"EQU")!=0) //if label is present in the instruction
                        //and its not a constant then its relative
     fprintf(f3,"%s\t\t\t%d\t\t\t%s\n",la,len1,"R");
   else if(strcmp(la,"X")!=0 && strcmp(mne,"EQU")==0) //if label is present in the instruction
                        //and its a constant then its absolute
     fprintf(f3,"%s\t\t\t%d\t\t\t%s\n",la,len1,"A");
```

```
}
//lookup function for psuedo op table
int potget()
{    char pop[20],proce[20];
     rewind(f4); int i=0;
     while(i<7)
     {i++;
       fscanf(f4,"%s%s",pop,proce);      //read opcode one by one
       if(strcmp(pop,mne)==0)
       return 1;//if mnemonic is found in pot
     }
     return 0; //if not found

}
char instlen[2];
char opcode[10],m[10],format[10];
//lookup function for machine op table
void motget()
{
     rewind(f2);
     for(j=0;j<11;j++)
     {
       fscanf(f2,"%s%s%s%s",m,opcode,instlen,format);      //read machine op table file line wise

       if(strcmp(m,mne)==0)      //if mnemonic is found in psuedo op table
       {
            //write to assembled file TO PASS1.TXT
            fprintf(f5,"%d\t\t\t%s\t\t\t%s\n\n",lc,opcode,opnd);
      //update location counter
            if(strcmp(instlen,"2")==0)
      lc+=2;
            else if(strcmp(instlen,"4")==0)
```

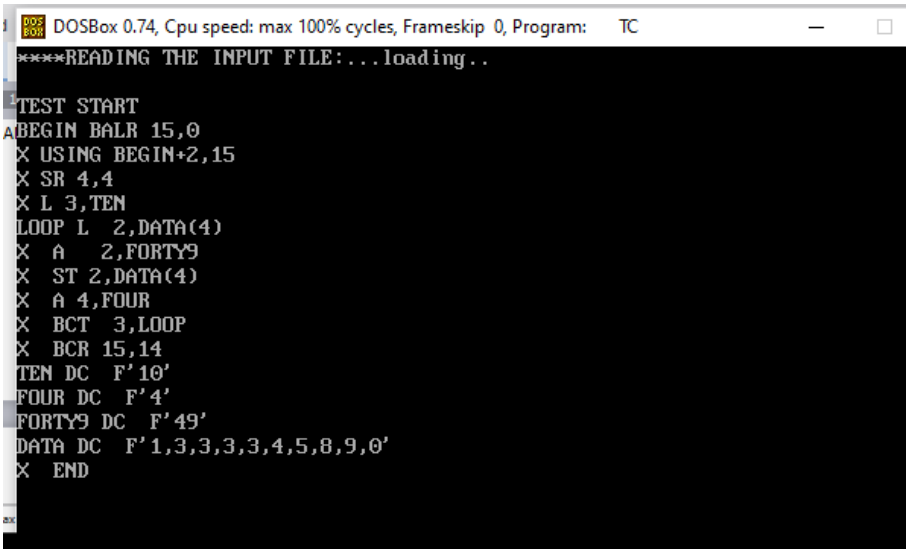
```
       lc+=4;
       break;
         }
        }

}
void readf(FILE *fp)
{
   rewind(fp);
   char ch=fgetc(fp);

   while(ch!=EOF)
   {
     //if(ch!='X')
     cout<<ch;      ch=fgetc(fp);
   }
   getch();
}
void passone()
{
     fscanf(f1,"%s%s",la,mne); //takes ist instr,stores its label in la, opcode mnemonic in mne
     strcpy(name,la);                //copies label to name
     lc=0;//sets location counter to 0 when ist instruction is encountered ,sets location counter
     fprintf(f3,"%s\t\t\t%s\t\t\t%s\n\n","symbol","length","Relocation");
     fprintf(f3,"%s\t\t\t%s\t\t\t%s\n",la,"1","R"); //add ist label to symbol table
     //f5=fopen("pass1.txt","w+");    //file in which pass 1 output will be stored

     fprintf(f5,"\n%s\t%s\t\t\t%s\n\n","RELATIVE_LOCATION","OPCODE","OPERANDS");
     //********now dealing with next instructions
     //getch();
     for(j=0;j<15;j++)
     {
```

```
fscanf(f1,"%s%s%s",la,mne,opnd);      //read instruction
addsymbol();//label on card;add it to symbol table
int flag=0;
if(strcmp(mne,"END")!=0)
flag=potget(); //search pseudoop table to check if the mnemonic is psuedo op      else    //if not a psuedo op add to machine op table
else
flag=1;
if(flag==0)
{
  motget(); //search machine op table;if not pseudo op look in machine op table
}
}//endwhile
//AFTER THE PRGRAM HAS BEEN PLACED IN MEMORY ,NOW ALL DC AND DS NEED TO BE
//STORED
rewind(f1);
fscanf(f1,"%s%s",la,mne);      //scan ist instruction
for(j=0;j<15;j++) //scan other instructions
{
  fscanf(f1,"%s%s%s",la,mne,opnd);      //read instruction

  if(strcmp(mne,"DC")==0)
  {
    int i=0;
    while(opnd[i++]!='\0')
    {
      if(( opnd[i]=='\'')||opnd[i]=='F')
   opnd[i]=' ';
    }
      //write to assembled file TO PASS1.TXT
        fprintf(f5,"%d\t\t\t%s\t\t\t%s\n\n",lc,mne,opnd);
 //update location counter
```

```
      lc+=4;
       }              }



}

void main()
{

     clrscr();
     f1=fopen("spp.txt","r"); //open the text file contaning program in read mode
     f3=fopen("symtab.txt","w+");    //table files to be created are opened in write mode
     f2=fopen("mopt.txt","r"); //open machine op table file in read mode
     f4=fopen("popt.txt","r"); //open psuedo op table file in read mode
     f5=fopen("pass1.txt","w+");    //file in which pass 1 output will be stored
     //f6=fopen("pass2.txt","w+");//file in which pass 2 output will be stored
     //call to pass one execution
     passone();
     //READ FILES ONE BY ONE                                        2
    cout<<"****READING THE INPUT FILE:...loading.."<<endl<<endl;
     readf(f1); //display original program
     fclose(f3);
     f3=fopen("symtab.txt","r");
     getch();
     ////////////////////////////////////////////////

     ////reading symboltable file//////////////////////
     clrscr();
     cout<<"****READING THE SYMBOL TABLE FILE:...loading.."<<endl<<endl;
     readf(f3);      //display symbol table
     getch();
     fclose(f3);
```

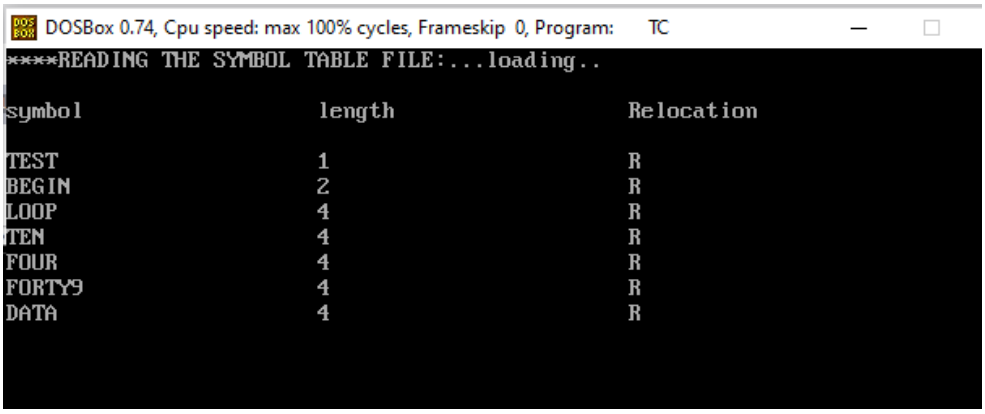
```
    ////////////////////////////////////////////////

    ///////display pass one assembled file//////////////
    clrscr();
    fclose(f5);
    f5=fopen("pass1.txt","r");
    cout<<"\n\n****READING THE ASSEMBLED PASS 1 FILE:...loading.."<<endl<<endl;
    readf(f5);      //display pass1 assembled file
    getch();
    fclose(f5);
    ////////////////////////////////////////////////

     getch();
}
```

Input file:

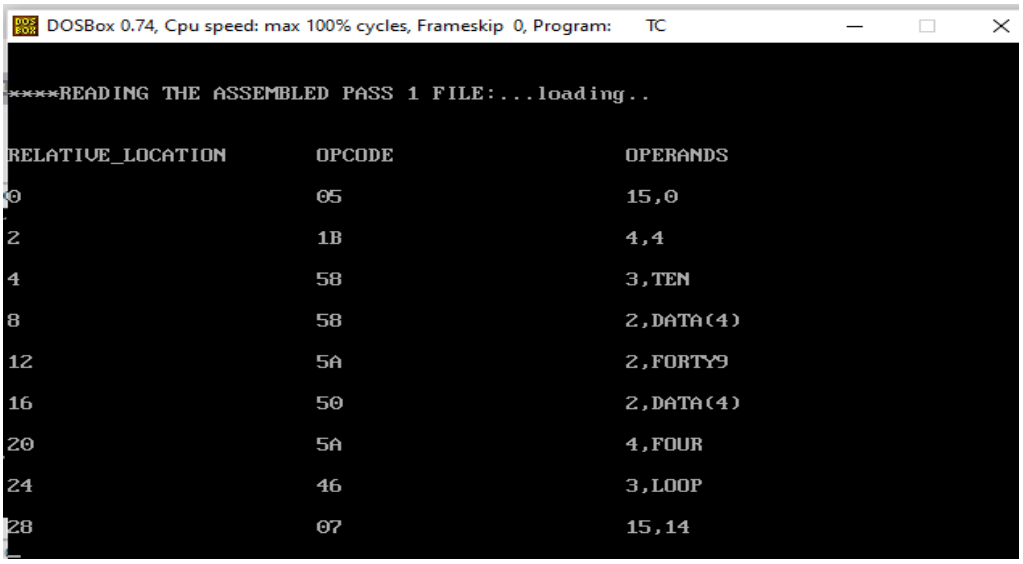


OUTPUT:

Symbol table:



OUTPUT OF PASS ONE:



**********PROGRAM 2**********

```
//Simulation of single pass macro processor for the given program to generate argument list array,MDT and MNT
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<iostream.h>
#include<string.h>
FILE *f1,*f2,*f3,*f4,*f5;//
int nmflag=0;//is set to 1 when there are nested macros
char lbl[10],arg1[10],arg2[10],arg3[10];
//this function reads the files
void readf(FILE *fp)
{
   //clrscr();
   rewind(fp);
   char ch=fgetc(fp);

   while(ch!=EOF)
   {
        cout<<ch;
        ch=fgetc(fp);
   }
   getch();
}
//this functon prepares argument list array
void arglist()
{ // clrscr();
   fprintf(f2,"%s\t\t\t\t\t%s\n","INDEX","ARGUMENT");//write header to ALA
   fscanf(f1,"%s",lbl); //fetch label of the instruction
      if(strcmp(lbl,"MACRO")==0)
          fscanf(f1,"%s%s%s%s",lbl,arg1,arg2,arg3);//fetch arguments and macroname
```

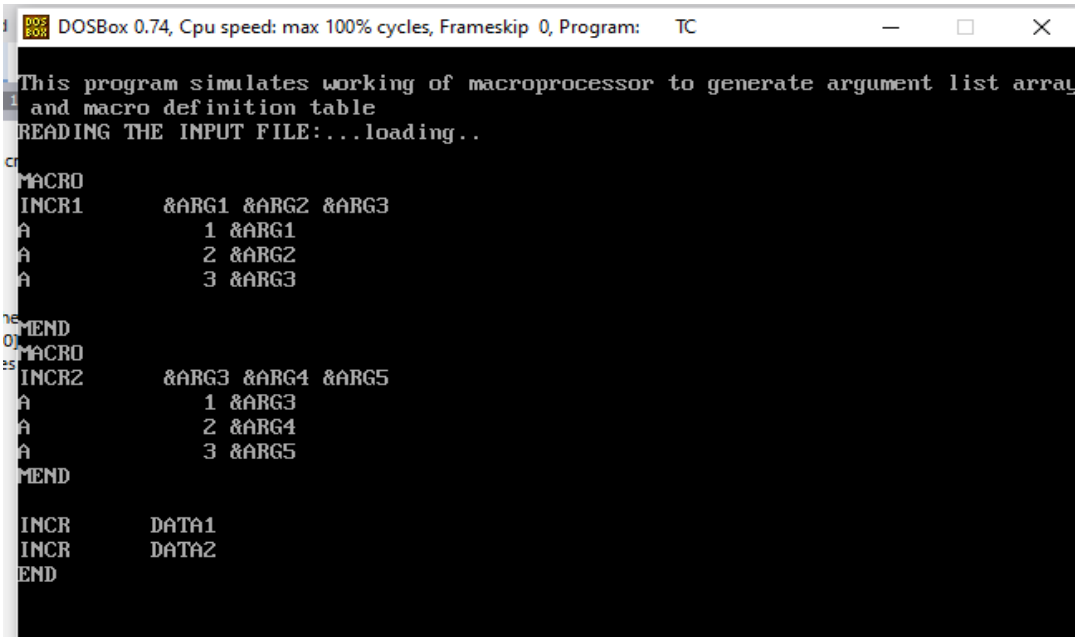
```
    fprintf(f2,"\n%d\t\t\t\t\t%s",1,arg1);    //write arguments to the ALA
    fprintf(f2,"\n%d\t\t\t\t\t%s",2,arg2);
    fprintf(f2,"\n%d\t\t\t\t\t%s",3,arg3);

}
void mnt()
{
      rewind(f1);
      int i=0;
      fprintf(f4,"%s\t\t\t\t\t%s\n","INDEX","MACRO NAME");//write header to ALA
      fscanf(f1,"%s",lbl); //fetch label of the instruction    --macro
      while(strcmp(lbl,"END")!=0)
      { //fscanf(f1,"%s",lbl);
        if(strcmp(lbl,"MACRO")==0){
            fscanf(f1,"%s",lbl);
            fprintf(f4,"\n%d\t\t\t\t\t%s",i++,lbl);//fetch arguments
            fscanf(f1,"%s",lbl);
              }
        else
         fscanf(f1,"%s",lbl);

        }
}
//create macro definition table
void create_mdt()
{
      clrscr();
      rewind(f1);
      look:
      fscanf(f1,"%s",lbl);
      if(strcmp(lbl,"MACRO")==0)
        {
```

```
        fscanf(f1,"%s%s%s%s",lbl,arg1,arg2,arg3); //picks ist line
        fprintf(f3,"\n%s\t\t\t\t\t%s,%s,%s",lbl,arg1,arg2,arg3);    //writes it in mdt
     int i=0;    //maintains count of macros

        while(strcmp(lbl,"MEND")!=0)
        {                       i++;
            fscanf(f1,"%s%s%s",lbl,arg1,arg2);
               if((strcmp(lbl,"MEND")!=0))
               {
          // else
                  fprintf(f3,"\n%s\t\t\t\t\t%s,#%d",lbl,arg1,i);
              fscanf(f1,"%s%s%s",lbl,arg1,arg2); }
                  if((strcmp(lbl,"MEND")!=0))
                fprintf(f3,"\n%s\t\t\t\t\t%s,#%d",lbl,arg1,i);
                  else
              fprintf(f3,"\n%s",lbl);
              }
      }   //end if
    else if(strcmp(lbl,"END")==0)
    return;
    else
        goto look;
}
void main()
{

    clrscr();
    cout<<"\nThis program simulates working of macroprocessor to generate argument list array and
macro definition table"<<endl;
    f1=fopen("macro.txt","r");//read input program
    f2=fopen("arglist.txt","w+"); //create argument list array filr
    f3=fopen("mdt.txt","w+");    //create macro def table file
```

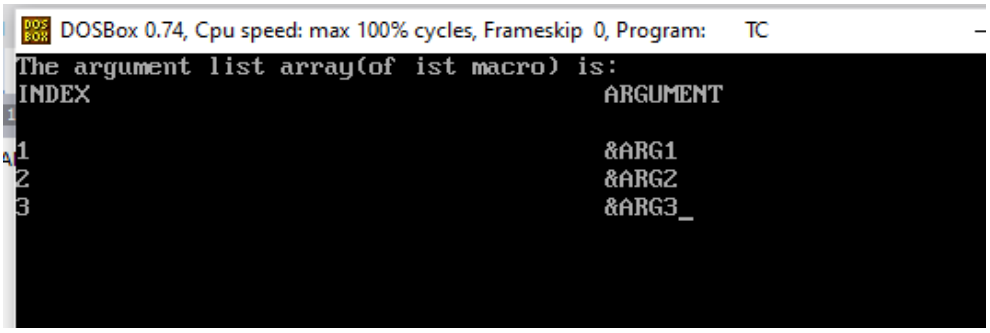
```
f4=fopen("mnt.txt","w+"); //create mnt file
arglist();          //call to function that creates argument list array
//READ FILES ONE BY ONE
//\clrscr();
//create macro name table
mnt();
cout<<"READING THE INPUT FILE:...loading.."<<endl;
readf(f1);
//displaying argument list array
clrscr();
cout<<"The argument list array(of ist macro) is:"<<endl;
readf(f2);
create_mdt();//create macro definition table
clrscr();
cout<<"The macro defintion table(of first macro) is :\n";
readf(f3);
fclose(f3);
clrscr();
cout<<"READ MACRO NAME TABLE FILE:\n";
readf(f4);
}
```

OUTPUT:

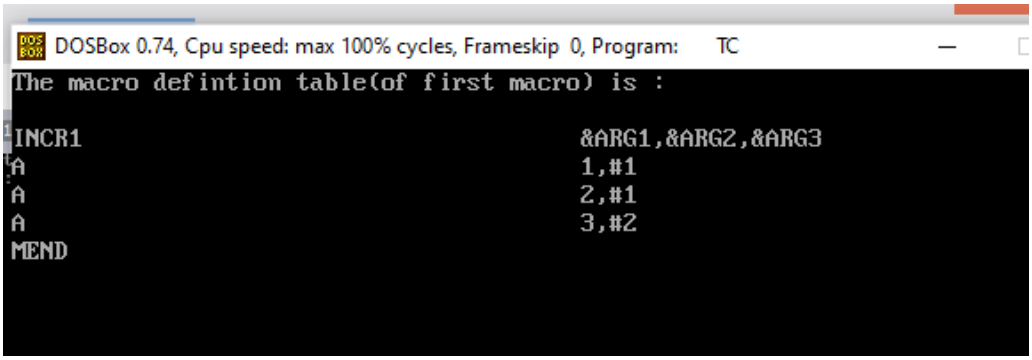
INPUT FILE:



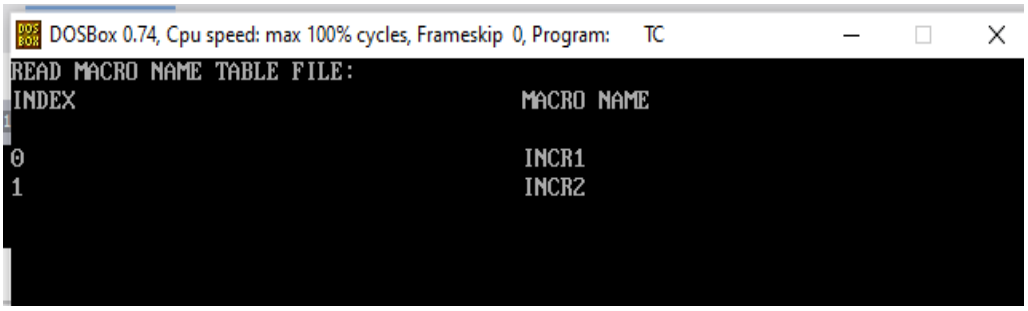
ARGUMENT LIST ARRAY:



MACRO DEFINITION TABLE:



MACRO NAME TABLE:



****