ASSEMBLER DESIGN FOR SIC/XE

Group ID-19

Team Members-

- T. Lokesh Reddy (17114073)
- Aman Verma (17114009)
- Abhijeet Shakya (17114002)
- G. Vishnuteja (17114031)
- Karan Lamba (17114043)

Problem Statement-

Design and implement a version of SIC/XE assembler yourself to demonstrate major functions of a two-pass SIC/XE assembler: Pass 1 and Pass 2. Consider all the SIC/XE instructions (given below) from the textbook as the instruction set for your assembler design and implementation. Please ensure to include Format 2, 3 and 4 instructions in the set.

Introduction and Background

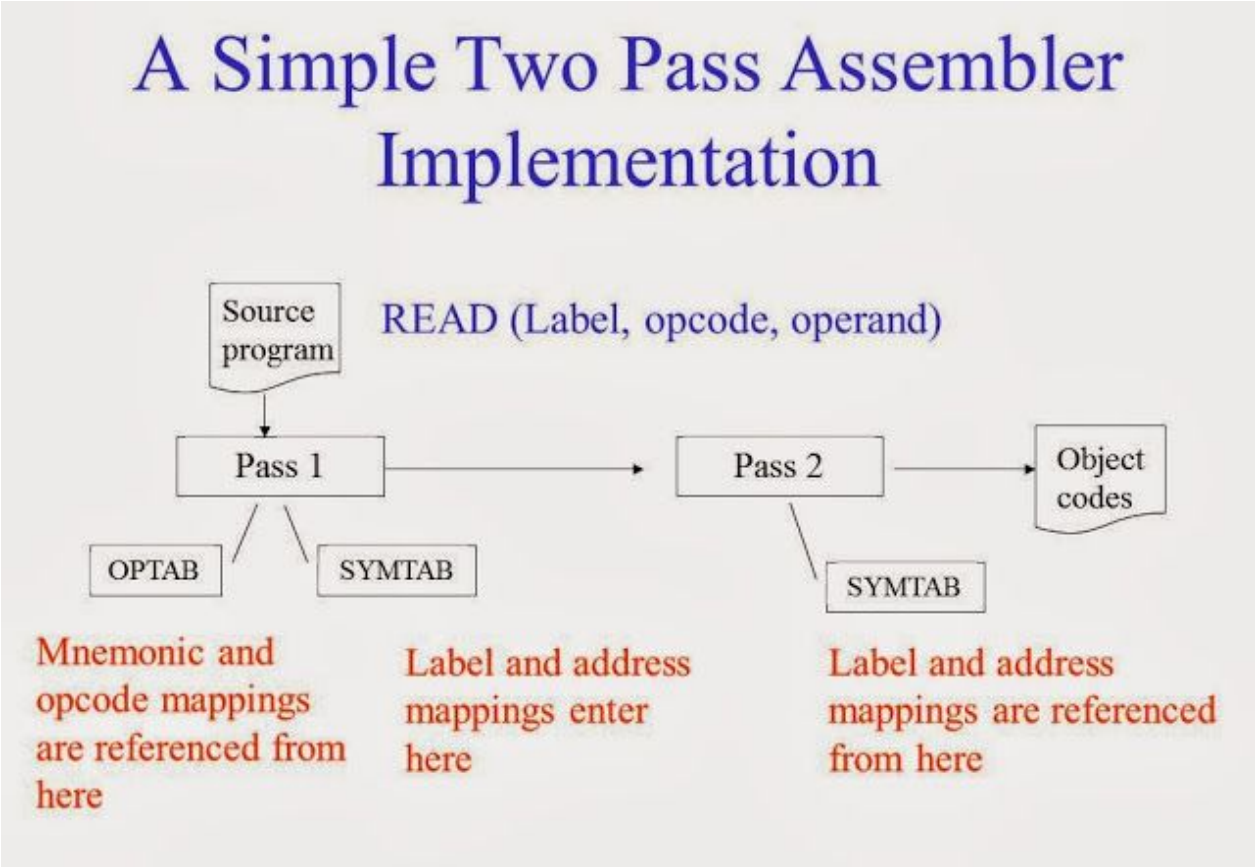
This is a SIC/XE Assembler with implementation of :

1. Basic Assembler
   a) Input Format free
   b) Error Handling

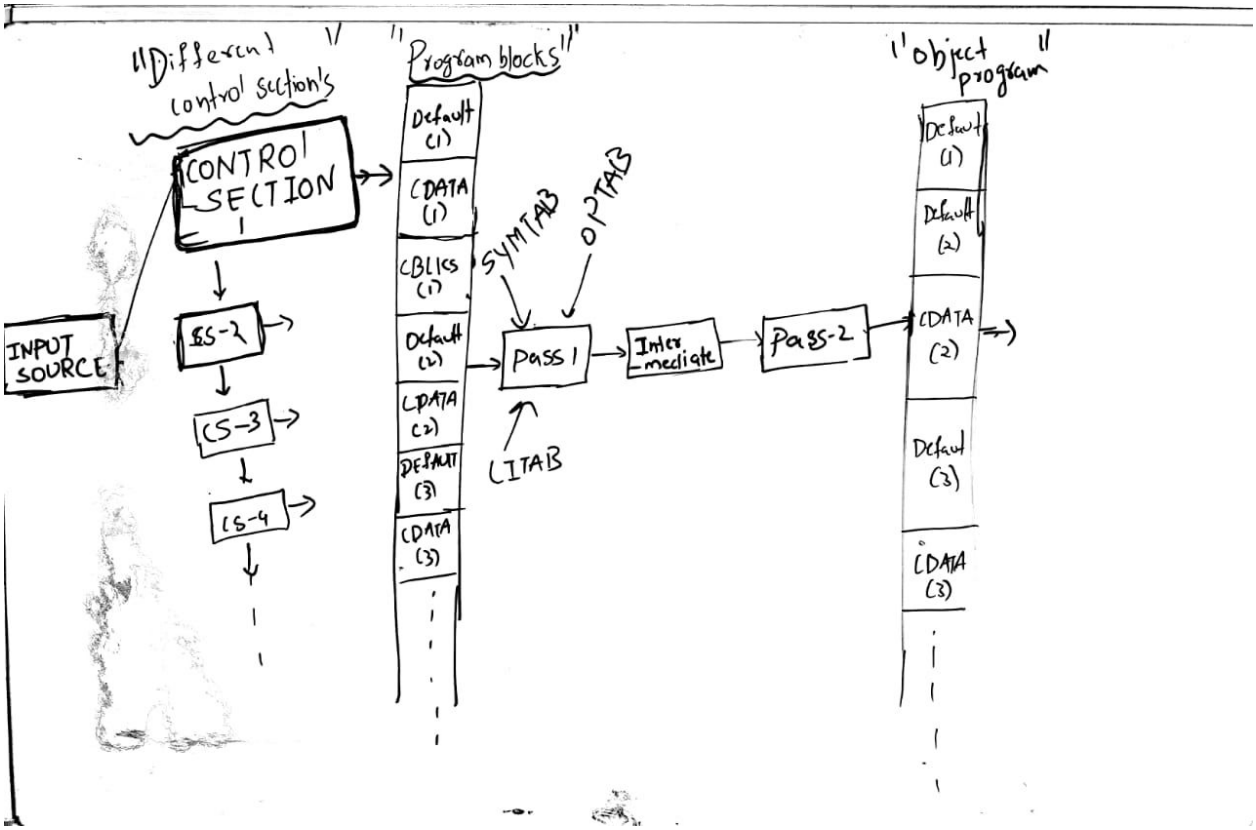
2. Machine Independent Functions
   a) Program Blocks
   b) Control Sections
   c) Use of Literals
   d) Assembler Directives such as EQU, LTORG

We are taking an input file through a .txt file and it will pass through Pass1.cpp. Consequently, it will form an intermediate file which will convert the instructions into a proper format which would be further used by Pass2.cpp to convert it into the object file (.txt).

Architecture and design of the software



Abstract Design of our Assembler



FLOW CHART OF THE DESIGN

Basically our design is structural rather than object oriented.So we implemented the architecture using various functions like Pass1 and Pass2.

The files which were used to build the assembler (architecture) :

- **Input.txt** - Contains various inputs

- **Nainput.txt** - Contains the source program to be processed by the Assembler

- **Main.cpp** - The actual code to be compiled which includes (import) all files like Conversions.cpp , DATA_STRUCTURES.cpp, Pass1.cpp and Pass2.cpp
  It divides the program in the Nainput.txt in accordance with various control sections and send the file to tempinput.txt and send it to Pass1.cpp

- **Pass1.cpp** - It takes the input from tempinput.txt and writes the sequence of instructions in proper format so as to make sure that Pass2.cpp can use it.

- **Intermediate.txt , Intermediate1.txt** - Intermediate.txt contains the output of Pass1.cpp for the corresponding tempinput.txt.To see the complete intermediate file for the complete input Nainput.txt , go through Intermediate1.txt

- **Pass2.cpp** - It will convert the intermediate file into the object program and save it in object.txt file.It will also show the flow of the program int the list.txt file.

- **Modifications.txt -** It contains all the modifications needed for tempinput.txt (program relocation).

- **Object.txt** - It contains the final object code for the **complete** input source program in Nainput.txt.

- **List.txt** - It contains the flowchart of the instructions along with the SYMTAB view and LITTAB view.

- **Conversions.cpp** - It contains the required functions used by Pass1.cpp and Pass2.cpp like converting decimal numbers to hexadecimal and vice versa .

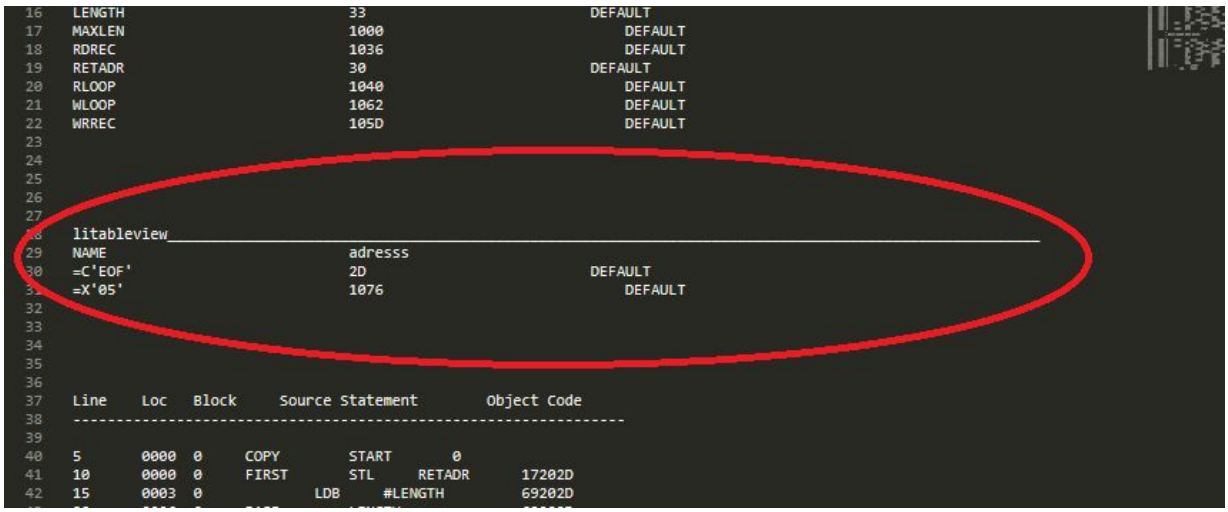
- **DATA_STRUCTURES.cpp**-
  It contains data structures like:
  a.) LITTAB
  b.) SYMTAB
  c.) OBTAB
  d.) EXPORT
  e.) IMPORT

The data contained in these data structures are temporary for the running control section and will be replaced by another control section variables and constants when that control section is being processed by Pass2.These data structures will be filled by Pass1.cpp and used by Pass2.cpp.

Algorithms and data structures used in the implementation

It contains data structures like:

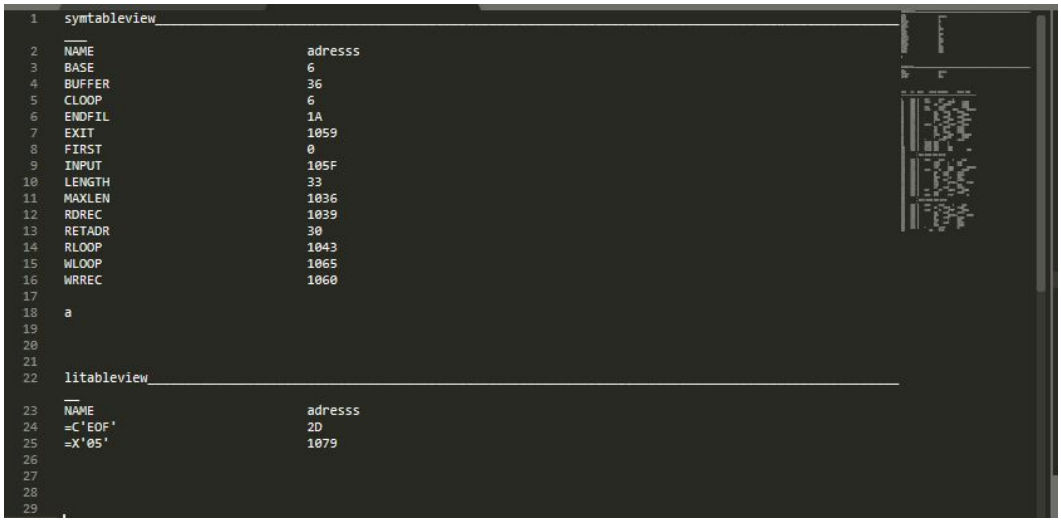
a.) LITTAB - It contains the literals and their corresponding addresses.



```
LENGTH                  33                      DEFAULT
MAXLEN                  1000                        DEFAULT
RDREC                   1036                        DEFAULT
RETADR                  30                      DEFAULT
RLOOP                   1040                        DEFAULT
WLOOP                   1062                        DEFAULT
WRREC                   105D                        DEFAULT



litableview_____________________________________________________________________________________
NAME                    adresss
=C'EOF'                 2D                      DEFAULT
=X'05'                  1076                        DEFAULT



Line    Loc   Block     Source Statement        Object Code
-----------------------------------------------------------------

5       0000  0     COPY        START       0
10      0000  0     FIRST       STL     RETADR      17202D
15      0003  0             LDB     #LENGTH         69202D
```

Here in every list file you can see Littabview

b.) SYMTAB - It contains all the variables and labels and related info.



```
symtableview_____________________________________________________________________________________
___
NAME                    adresss
BASE                    6
BUFFER                  36
CLOOP                   6
ENDFIL                  1A
EXIT                    1059
FIRST                   0
INPUT                   105F
LENGTH                  33
MAXLEN                  1036
RDREC                   1039
RETADR                  30
RLOOP                   1043
WLOOP                   1065
WRREC                   1060

a



litableview______________________________________________________________________________________
__
NAME                    adresss
=C'EOF'                 2D
=X'05'                  1079
```

Similar to litab view you can see the symtable also

c.) OBTAB - It has the opcodes for the pneumonic instructions and other info like format number.

d.) EXPORT - It contains all the variables and constants which has to be exported to other control sections.

e.) IMPORT - It contains all the variables and constants which has to be used by the running control section which are defined in other control sections.

Algorithm for MAIN.cpp

1.Begin
2.Take control on Nainput.txt to read similarly on tempinput.txt to write
3.clear the files object.txt,tempinput.txt,intermediate.txt,intermediate1.txt
4 create string CSECT
5 while(Nainput.txt contain lines)
  {
    1'.Read the line;
    2'.if(line contains "start")initialize csect to program name and copy to tempinput.txt
   3'.if(line contains "csect")
        {
            Send tempinput.txt to pass1;
            pass1();
            pass2();
            Clear the tempinput.txt
            Change csect variable.
        }
   4' .write line to temp.input.txt;
   5'.if(line contains end)
     {

```
        pass1();
        pass2();
        Break;
    }
```

Algorithm for pass1

BEGIN {construction of symbol table}
Skip over initial comment lines
Process the START statement, if present,
setting Locctr and ENDval to the operand's value
Loop through the source lines until the END
statement is reached or the source file runs out
BEGIN
Skip over source lines that are comment lines
Extract Label, Opcode, & Operand parts
IF there is a Label
add it to the symbol table if it is not
already there (otherwise it's an error)
Increment Locctr:
1. If Opcode is a storage directive, for
BYTE, RESB, or RESW, Operand determines
increment; for WORD it is 3
2. If Opcode is an instruction, the increment

is 1,2,3, or 4 as per Opcode bits
3. Increment = 0 for assembler directives
END {of loop}
If present, process END statement, and reset ENDval to the END statement's operand value, if present
END {of Pass 1}

Algorithm for pass2

BEGIN {generation of object module}
Write assembler report headings & any leading comment lines (Note: as each source line is processed, it is written to the assembler report)
Process the START statement, if present, setting Locctr to the operand's value (default is 0)
Initialize the object module:
1. Locctr value is initial load point
2. ENDval from Pass 1 is tentative "execute next"
Loop through the source lines until the END statement is reached or source runs out
BEGIN
Skip over any comment lines (but write them to the assembler report)
Extract Opcode, & Operand, increment Locctr, then if Opcode is

1. RESW or RESB, start a new module:
a. ! delimiter to end prior module
b. loader address replaces ENDval in
 prior module as "execute next"
c. Locctr value is next load point
d. ENDval from Pass 1 as this module's
a tentative "execute next"
2. WORD or BYTE, Operand gives the storage
value(s) to write to the object module
3. an assembler directive, process as spec'd
4. an instruction, build the object version
utilizing nixbpe bits, Locctr, and
Operand value from the symbol table
END {of loop}
Append the ! delimiter to end the final module
Output the object module(s) as the object code file
if no errors were encountered in Pass 1 or 2
END {of Pass 2}

Design Results

The list.txt file shows the design results for every source input program which determines the object file and also the flowchart of the instructions along with

the Symtabview and Littableview.The flowchart contains the info of each instruction like

1. Location
2. Block Number
3. Labels and Label Field Variables
4. Opcodes for that instruction
5. Address Field Variables

In the design of this project we also included the error file to know if any error occurred during the computation.

Following are the results expected while implementing this design :

- It should be input format free.
- It should be able to handle if any errors occur and display the associated error in the error.txt file.
- It should handle the assembler directives such as **USE,LTORG,CSECT,EQU** along with basic assembler instructions.
- It should load instructions in various blocks if required.i.e. It should handle the **program blocks.**

- It should take instructions from different control sections and use the variables defined in other control sections since we have designed the IMPORT and EXPORT data structures.In this way it handles the **control sections** block .
- It should display the object file along with the modifications required in the program.

```
list.txt                    inputkyahe.txt                listfile.txt
Line    Loc    Block     Source Statement        Object Code
-----------------------------------------------------------------

5       0000   0     COPY        START       0
10      0000   0     FIRST       STL     RETADR      172063
15      0003   0     CLOOP       JSUB        RDREC       4B2021
20      0006   0             LDA     LENGTH      032060
25      0009   0             COMP        #0          290000
30      000C   0             JEQ     ENDFIL      332006
35      000F   0             JSUB        WRREC       4B203B
40      0012   0             J       CLOOP       3F2FEE
45      0015   0     ENDFIL      LDA     =C'EOF'     032055
50      0018   0             STA     BUFFER      0F2056
55      001B   0             LDA     #3          010003
60      001E   0             STA     LENGTH      0F2048
65      0021   0             JSUB        WRREC       4B2029
70      0024   0             J       @RETADR         3E203F
75  0000  1            USE     CDATA
80      0000   1     RETADR  RESW        1
85      0003   1     LENGTH  RESW        1
90  0000  2            USE     CBLKS
95      0000   2     BUFFER  RESB        4096
100     1000   2     MAXLEN  WORD        4096            1000
105            .
110            . READ RECORD INTO BUFFER
115            .
120 0000  0            USE     DEFAULT
125     0027   0     RDREC       CLEAR       X       B410
130     0029   0             CLEAR       A       B400
135     002B   0             CLEAR       S       B440
140     002D   0             +LDT        #4096           75001000
145     0031   0     RLOOP       TD      INPUT       E32038
150     0034   0             JEQ     RLOOP       332FFA
155     0037   0             RD      INPUT       DB2032
160     003A   0             COMPR       A,S     A004
165     003C   0             JEQ     EXIT        332008
170     003F   0             STCH        BUFFER,X        57A02F
175     0042   0             TIXR        T       B850
180     0044   0             JLT     RLOOP       3B2FEA
185     0047   0     EXIT        STX     LENGTH      13201F
190     004A   0             RSUB                4F0000
Line 95, Column 1
```

View of list.txt (Design result)

Source code

How to compile ?

1. Put all the source code files in one folder. (**must**)
2. Take an input from input.txt /user input and copy it to Nainput.txt.
3. Compile Main.cpp only using any compiler and see the output in the object.txt file and flowchart in the list.txt file.
4. See the errors in errors.txt if encountered.

You can find the related source code files in the zip attachments. Here we are providing the code in small font size :) **Move to PAGE 36 to skip the src code.**

MAIN.cpp

```
#include<bits/stdc++.h>
#include<iostream>
#include<cstdlib>
#include<fstream>
#include<string>
#include<cstdio>
#include<map>
#include<climits>
#include "Pass2.cpp"

void clean()
{
	fstream intm3;intm3.open("intermediate1.txt",ios::out | ios::trunc); intm3.close();
	fstream intm31;intm31.open("list.txt",ios::out | ios::trunc); intm31.close();
```

```
        fstream intm32;intm32.open("object.txt",ios::out | ios::trunc); intm32.close();
        fstream intm34;intm32.open("tempinput.txt",ios::out | ios::trunc); intm34.close();

}



int main()
{
        fstream fin;
        fstream fout;

        fin.open("Nainput.txt");
        fout.open("tempinput.txt", ios::out | ios::trunc);


        string CSECT;

        clean();

        while(1)
        {

                string s;
                getline(fin,s);

                if(s.find("START")!=string::npos)
                {
                        int i=0;
                        CSECT="";
                        while(s[i]!=' '){CSECT+=s[i];i++;}
                }


                if(s.find("CSECT")!=string::npos)
                {
                         fout<<"END"<<"\t"<<CSECT<<endl;
                          int i=0;
                        CSECT="";
                        while(s[i]!=' '){CSECT+=s[i];i++;}
                       run2();  //pass1();pass2();

                        fout.close();
                        fout.open("tempinput.txt", ios::out | ios::trunc);


                }


                if(s.find(" END ")!=string::npos)
                {


                        string h1=s;
        string s1;
        getline(fin,s1);
```

```
        if(s1.find("*")!=string::npos)
        {
           fout<<s1<<endl;
        }

       fout<<h1<<endl;
                        run2();  pass1();pass2();
                        break;



                      }
     fout<<s<<endl;




          }



}
```

Pass1.cpp

```
#include<iostream>
#include<cstdlib>
#include<fstream>
#include<string>
#include<cstdio>
#include<map>

using namespace std;

#include "DATA_STRUCTURES.cpp"
#include "Conversions.cpp"

bool isWhiteSpace(char a)
{
   if(a==' ')  return true;
   if(a=='\t') return true;
   return false;
}

void printforview()
{




   if(1)
```

```
    {


      fstream   lst;
      lst.open("list.txt",ios::out | ios::app);
      lst<<""<<endl;
       lst<<""<<endl;
       lst<<""<<endl;
       lst<<""<<endl;
       lst<<""<<endl;

lst<<"symtableview"<<"_____________________________________________________________________________________________________________"<<endl;
      lst<<"NAME\t\t\t\t\t\t\tadresss\t\t\t\t\t\t\tBlock"<<endl;
      MapType3::iterator it;

      for (it = SYMTAB.begin(); it != SYMTAB.end(); it++)
       {
           pair<string,info_label> tra=*it;
           if(tra.second.exist=='n')continue;
           if(tra.first[0]=='=')continue;
           lst<<tra.first<<"\t\t\t\t\t\t\t"<<tra.second.address<<"\t\t\t\t\t\t\t"<<tra.second.block<<endl;

       }




       lst<<""<<endl;
       lst<<""<<endl;
       lst<<""<<endl;
       lst<<""<<endl;
       lst<<""<<endl;


       lst.close();

    }
    if(1)
    {


      fstream   lst;
      lst.open("list.txt",ios::out | ios::app);

lst<<"litableview"<<"_____________________________________________________________________________________________________________"<<endl;
      lst<<"NAME\t\t\t\t\t\t\tadresss"<<endl;
      MapType7::iterator it;

      for (it = LITAB.begin(); it != LITAB.end(); it++)
       {
           pair<string,info_litab> tra=*it;
           if(tra.second.exist=='n')continue;
          // if(tra.first[0]=='=')continue;
           lst<<tra.first<<"\t\t\t\t\t\t\t"<<tra.second.address<<"\t\t\t\t\t\t\t"<<tra.second.block<<endl;
```

```
        }


        lst<<""<<endl;
        lst<<""<<endl;
        lst<<""<<endl;
        lst<<""<<endl;
        lst<<""<<endl;

        lst.close();




    }









}

void extract(string a,string word[],int& count)
{
    int i;
    for(i=0;i<5;++i) word[i]="";
    count=0;
    i=0;
    while(isWhiteSpace(a[i])&&i<a.length()) {++i;continue;}
    if(i==a.length()||a[i]=='.')   return;
    for(;i<a.length();)
    {
        while(isWhiteSpace(a[i])&&i<a.length()) {++i;continue;}
        if(i==a.length()) break;
        for(;!isWhiteSpace(a[i])&&i<a.length();++i) word[count]+=a[i];
        ++count;
    }

    if(word[0]!="EXTDEF")
    {
        if(word[0]!="EXTREF")
        {
            return;
        }
    }


    string h1=word[1];
    count--;
    word[1]="";
    for(int k1=0;k1<h1.length();k1++)
    {
        if(!isWhiteSpace(h1[k1])&&h1[k1]!=',')
        {
            word[count]+=h1[k1];
```

```
        }
        if(h1[k1]==',')count++;
    }

    count++;





}

void execute(string[],int);

int block_num,line;
hexa pc;
string curr_block;
bool error_flag=0;


ifstream fin1;
ofstream fout1,error;
fstream naku;


void run()
{

    fin1.close();
    fout1.close();
    error.close();



    string cset1;
    SYMTAB.clear();
    OPTAB.clear();
    SYMTAB.clear();
    BLOCK.clear();
    IMPORT.clear();
    EXPORT.clear();
    create();
    char ch;
    string s,word[5];
    int count=0;
    fin1.open("tempinput.txt");
    fout1.open("intermediate.txt",ios::out | ios::trunc);
    error.open("error.txt");
    line=5;
    getline(fin1,s);
    extract(s,word,count);
    while(count==0)
    {
        fout1<<line<<endl;
        fout1<<"$";
        fout1<<s<<endl;
```

```
    fout1<<""<<endl;
    fout1<<""<<endl;
    line+=5;
    cout<<"s: "<<s<<endl;
}
pc="0";
BLOCK["DEFAULT"].num=0;
BLOCK["DEFAULT"].address=pc;
BLOCK["DEFAULT"].length="0";
BLOCK["DEFAULT"].exist='y';
curr_block="DEFAULT";
block_num=1;
line=5;
if(word[0]=="START")
{
    pc=word[1];
    fout1<<line<<endl;
    fout1<<""<<endl;
    fout1<<"START"<<endl;
    fout1<<pc<<endl;
    fout1<<pc<<endl;
    fout1<<""<<endl;
    cout<<"0 is start!"<<endl;
}
else if(word[1]=="START")
{
    pc=word[2];
    fout1<<line<<endl;
    fout1<<word[0]<<endl;cset1=word[0];
    fout1<<"START"<<endl;
    fout1<<pc<<endl;
    fout1<<pc<<endl;
    fout1<<""<<endl;
    cout<<"1 is start!"<<endl;
}
else if(word[1]=="CSECT")
{
    fout1<<line<<endl;
    fout1<<word[0]<<endl;cset1=word[0];
    fout1<<"START"<<endl;
    fout1<<pc<<endl;
    fout1<<pc<<endl;
    fout1<<""<<endl;
    cout<<"1 is start!"<<endl;
}
else
    execute(word,count);
while(true)
{
    getline(fin1,s);
    if(s.find("EXTDEF")!=string::npos||s.find("EXTREF")!=string::npos)
    {
       extract(s,word,count);
       for(int i=1;i<count;i++)
       {
         if(word[0]=="EXTDEF")
         {
```

```
            EXPORT[word[i]].exist='y';
            EXPORT[word[i]].cset=cset1;
        }
        else
        {


           IMPORT[word[i]].exist='y';
           IMPORT[word[i]].cset=cset1;
        }
      }
      continue;
    }
    if(s.find("LTORG")!=string::npos)continue;



    extract(s,word,count);
   // cout<<s<<endl;

    line+=5;
    cout<<"s: "<<s<<endl;
    fout1<<line<<endl;
    if(count==0)
    {
        cout<<"Comment detected!"<<endl;
        fout1<<"$"<<endl;
        fout1<<s<<endl;
        fout1<<""<<endl;
        fout1<<""<<endl;
        fout1<<""<<endl;
        continue;
    }

     if(word[0]=="END")
    {
        cout<<"entered
here**********************************************************************88for"<<cset1<<endl;
        BLOCK[curr_block].length=pc;
        fout1<<""<<endl;
        fout1<<word[0]<<endl;
        fout1<<word[1]<<endl;
        fout1<<pc<<endl;
        fout1<<""<<endl;

        printforview();



        break;

    }
    execute(word,count);
   // cin>>ch;
  }
  hexa addr,len;
  string temp=find_block(0);
```

```
    addr=BLOCK[temp].address;
    len=BLOCK[temp].length;
    for(int i=1;i<block_num;++i)
    {
        temp=find_block(i);
        BLOCK[temp].address=toHex(toDec(addr)+toDec(len));
        addr=BLOCK[temp].address;
        len=BLOCK[temp].length;

    }









}

void execute(string word[],int count)
{
    cout<<"word[0]: "<<word[0]<<" pc: "<<pc<<endl;
    if(word[0]=="USE")
    {
        cout<<"USE detected!"<<endl;
        BLOCK[curr_block].length=pc;
        if(word[1]=="")
        {
            word[1]="DEFAULT";
            curr_block="DEFAULT";
            pc=BLOCK["DEFAULT"].length;
        }
        else if(BLOCK[word[1]].exist=='y')
        {
            curr_block=word[1];
            pc=BLOCK[word[1]].length;
        }
        else
        {
            BLOCK[word[1]].num=block_num;
            BLOCK[word[1]].exist='y';
            BLOCK[word[1]].length="0";
            curr_block=word[1];
            pc="0";
            ++block_num;
        }
        fout1<<""<<endl;
        fout1<<word[0]<<endl;
        fout1<<word[1]<<endl;
        fout1<<pc<<endl;
        fout1<<""<<endl;
        return;
    }
    if(word[0][0]=='+')
    {
        cout<<"Format 4"<<endl;
        fout1<<""<<endl;
```

```
    fout1<<word[0]<<endl;
    fout1<<word[1]<<endl;
    fout1<<pc<<endl;
    pc=toHex(toDec(pc)+4);
    fout1<<pc<<endl;
    return;
}
if(OPTAB[word[0]].exist=='y')
{
    cout<<"0 is opcode!"<<endl;
    fout1<<""<<endl;
    fout1<<word[0]<<endl;
    fout1<<word[1]<<endl;
    fout1<<pc<<endl;
    pc=toHex(toDec(pc)+OPTAB[word[0]].format);
    fout1<<pc<<endl;
    return;
}
if(OPTAB[word[0]].exist=='n')
{
    if(SYMTAB[word[0]].exist=='y')
    {
        error<<"Line "<<line<<": Duplicate Symbol"<<endl;
        error_flag=1;
    }
    else
    {
        if(EXPORT[word[0]].exist=='y')
        {
           EXPORT[word[0]].address=pc;
        }


        if(word[0]=="*")
        {
            word[0]=word[1];
            word[2]=word[1].substr(1);
            word[1]="BYTE";
            LITAB[word[0]].address=pc;
            LITAB[word[0]].exist='y';
            LITAB[word[0]].block=curr_block;

        }
        if(word[1]=="EQU")
        {

             SYMTAB[word[0]].block=curr_block;
             SYMTAB[word[0]].exist='y';

             if(word[2]=="*")
             {
                SYMTAB[word[0]].address=pc;
                SYMTAB[word[0]].block=curr_block;

             }
             else
             {
```

```
        string h1=word[2];
        word[2]="";
        word[3]="";

       int  count =2;
       for(int k1=0;k1<h1.size();k1++)
       {
           if(h1[k1]=='-')count++;
           else
           {
              word[count]+=h1[k1];
           }

       }

       if(count==2)
       {
          SYMTAB[word[0]]=SYMTAB[word[2]];
          SYMTAB[word[0]].block=curr_block;
       }
       else
       {
           SYMTAB[word[0]].address=toHex(toDec(SYMTAB[word[2]].address)-toDec(SYMTAB[word[3]].address));
       }

       word[2]=h1;
    }


    fout1<<word[0]<<endl;
    fout1<<word[1]<<endl;
    fout1<<word[2]<<endl;
    fout1<<pc<<endl;
    fout1<<pc<<endl;


  return ;

}



SYMTAB[word[0]].address=pc;
SYMTAB[word[0]].block=curr_block;
SYMTAB[word[0]].exist='y';
fout1<<word[0]<<endl;
fout1<<word[1]<<endl;
fout1<<word[2]<<endl;
fout1<<pc<<endl;
if(word[1][0]=='+')
   pc=toHex(toDec(pc)+4);
else if(OPTAB[word[1]].exist=='y')
   pc=toHex(toDec(pc)+OPTAB[word[1]].format);
else if(word[1]=="WORD")      pc=toHex(toDec(pc)+3);
else if(word[1]=="RESW")      pc=toHex(toDec(pc)+(atoi(word[2].c_str())*3));
else if(word[1]=="RESB")      pc=toHex(toDec(pc)+atoi(word[2].c_str()));
```

```
        else if(word[1]=="BYTE")
        {

            int len=word[2].length()-3;
            if(word[2][0]=='X') len/=2;
            pc=toHex(toDec(pc)+len);
        }
        else
        {
           error<<"Line "<<line<<": Opcode not found"<<endl;
           error_flag=1;
        }
        fout1<<pc<<endl;
      }
   }
}
```

Pass2.cpp

```
#include<bits/stdc++.h>
#include<iostream>
#include<cstdlib>
#include<fstream>
#include<string>
#include<cstdio>
#include<map>
#include<climits>

using namespace std;

#include "Pass1.cpp"

ofstream obj,lst,mod;
ifstream intm;
int curr_block_num;

void modify_object_file()
{
   ifstream fin;
   fin.open("modification.txt");
   string s;
   while(true)
   {
      getline(fin,s);
      if(s==="") break;
      obj<<s<<endl;
   }
   fstream intm3;intm3.open("modification",ios::out | ios::trunc); intm3.close();
}
void copy()
{
    fstream transer;transer.open("intermediate1.txt",ios::out | ios::app);
    fstream intm2;intm2.open("intermediate.txt");
```

```
    transer<<intm2.rdbuf()<<endl;
 fstream intm3;intm3.open("intermediate.txt",ios::out | ios::trunc); intm3.close();
}

bool imm,ind;

void input(string a[])
{
    int i;
    for(i=0;i<6;++i)
        getline(intm,a[i]);
    cout<<"Input for line: "<<a[0]<<endl;
    for(i=1;i<6;++i)
        cout<<a[i]<<endl;
}

void assemble(string[]);
string gen_code(string[]);

string text_s="",text_e="";
int text_length=0,base=INT_MAX;

void run2()
{

    run();
    cout<<"entered
pass2_____________________________________________________________________________________________________"<<endl;
    string a[6];
    char ch;
    hexa start;
    if(0)
    {
        cout<<"Errors encountered! Listing file not prepared!"<<endl;
        cout<<"Have a look at the error file to know more!"<<endl;
        exit(1);
    }
    intm.open("intermediate.txt");
    obj.open("object.txt",ios::out | ios::app);

    lst.open("list.txt",ios::out | ios::app);
    mod.open("modification.txt",ios::out | ios::trunc);
    lst<<"Line\tLoc   Block\t\tSource Statement\t\tObject Code"<<endl;
    lst<<"-----------------------------------------------------------------"<<endl<<endl;
    input(a);
    curr_block="DEFAULT";
    curr_block_num=0;
    while(a[1]=="$")
    {
        lst<<a[0]<<"\t\t\t "<<a[2]<<endl;
        input(a);
    }
    if(a[2]=="START")
    {
        lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"  "<<curr_block_num<<"\t\t"<<a[1]<<"\t\t"<<a[2]<<"\t\t"<<a[3]<<endl;
        obj<<"H^";
```

```
 int i;
 for(i=0;i<a[1].length();++i)
    obj<<a[1][i];
 for(;i<6;++i)
    obj<<" ";
 string temp=find_block(block_num-1);
 hexa len;
 len=toHex(toDec(BLOCK[temp].address)+toDec(BLOCK[temp].length));
 obj<<"^"<<extendTo(6,a[3])<<"^"<<extendTo(6,len)<<endl;
 start=a[3];

MapType6::iterator it;
int x=0;
for (it = EXPORT.begin(); it != EXPORT.end(); it++)
{
    pair<string,info_define> tra=*it;
    if(tra.second.exist=='y')x++;

}
if(x>0)
{
  obj<<"D^";


    for (it = EXPORT.begin(); it != EXPORT.end(); it++)
     {
      pair<string,info_define> tra=*it;
      if(tra.second.exist=='n')continue;
      for(i=0;i<tra.first.length();++i)obj<<tra.first[i];for(;i<6;++i)obj<<" ";obj<<"^";
    obj<<extendTo(6,tra.second.address) ;
     }


  obj<<""<<endl;


}

x=0;
MapType5::iterator it1;
for (it1 = IMPORT.begin(); it1 != IMPORT.end(); it1++)
{
    pair<string,info_refernce> tra=*it1;
     if(tra.second.exist=='y')x++;


}
if(x>0)
{
  obj<<"R^";

   for (it1 = IMPORT.begin(); it1 != IMPORT.end(); it1++)
   {
    pair<string,info_refernce> tra=*it1;
    if(tra.second.exist=='n')continue;
    for(i=0;i<tra.first.length();++i)obj<<tra.first[i];for(;i<6;++i)obj<<" ";obj<<"^";
     obj<<extendTo(6,tra.second.address)<<"^";
```

```
        }

      obj<<""<<endl;

     }



    }
    else
    {
       error_flag=1;
       error<<"Intermediate File has no start!"<<endl;
    }
    while(true)
    {
      //
cout<<"*****************************************************************************************************
*************************************"<<endl;
       input(a);
       if(a[1]=="$")
       {
          lst<<a[0]<<"\t\t\t "<<a[2]<<endl;
          continue;
       }
       if(a[2]=="END")
       {

          lst<<a[0]<<"\t\t\t\t\t\t"<<a[2]<<"\t\t"<<a[3]<<endl;
          if(text_length>0)
          {
             obj<<text_s<<"^"<<extendTo(2,toHex(text_length/2))<<text_e<<endl;
             cout<<"!!"<<endl;
          }
          text_length=0;
          text_s="";
          text_e="";
          modify_object_file();
          obj<<"E^"<<extendTo(6,start)<<endl;

          copy();
          intm.close();
          lst.close();
          obj.close();
          mod.close();
         cout<<"endedone_____________________________________________________________";
          break;
       }
```

```
        assemble(a);
        cout<<"opcode: "<<a[2]<<":::";

    }
    if(error_flag)
    {
        cout<<"Errors encountered! Listing file not prepared!"<<endl;
        cout<<"Have a look at the error file to know more!"<<endl;
    }
    else
    {
        cout<<"INPUT FILE ASSEMBLED SUCCESSFULY!!"<<endl;
    }


}


void assemble(string a[])
{
    string object_code;
    hexa loc_ctr;
    int format;
    if(a[2]=="USE")
    {
        curr_block=a[3];
        curr_block_num=BLOCK[curr_block].num;
        cout<<curr_block<<"                        SDFGHJKLKJHGFDSDFGHUIOP:LKJHGFDXDFGHJKLKJHGFCX"<<endl;
        lst<<a[0]<<"\t0000  "<<curr_block_num<<"\t\t\t"<<a[2]<<"\t\t"<<a[3]<<endl;
        if(text_length>0) obj<<text_s<<"^"<<extendTo(2,toHex(text_length/2))<<text_e<<endl;
        text_s="";
        text_e="";
        text_length=0;
        return;
    }
    if(a[2]=="EQU")
    {
        lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"  "<<curr_block_num<<"\t\t"<<a[1]<<"\t"<<a[2]<<"\t\t"<<a[3]<<endl;
        return;

    }
    if(a[2]=="RESB"||a[2]=="RESW")
    {
        lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"  "<<curr_block_num<<"\t\t"<<a[1]<<"\t"<<a[2]<<"\t\t"<<a[3]<<endl;
        if(text_length>0) obj<<text_s<<"^"<<extendTo(2,toHex(text_length/2))<<text_e<<endl;
        text_s="";
        text_e="";
        text_length=0;
        return;
    }
    imm=ind=false;
    object_code=gen_code(a);
    cout<<"a[2]: "<<a[2]<<":::"<<object_code<<endl;
    if(a[1][0]=='=')
    {
     lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"  "<<curr_block_num<<"\t\t"<<'*'<<"\t\t\t"<<a[1]<<"\t\t\t"<<object_code<<endl;
```

```
    }
    else if((a[2]=="BYTE"||a[2]=="WORD"))
        lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"
"<<curr_block_num<<"\t\t"<<a[1]<<"\t"<<a[2]<<"\t\t"<<a[3]<<"\t\t\t"<<object_code<<endl;
    else
    {
        if(imm)      lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"
"<<curr_block_num<<"\t\t"<<a[1]<<"\t\t"<<a[2]<<"\t\t#"<<a[3]<<"\t\t\t"<<object_code<<endl;
        else if(ind) lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"
"<<curr_block_num<<"\t\t"<<a[1]<<"\t\t"<<a[2]<<"\t\t@"<<a[3]<<"\t\t\t"<<object_code<<endl;
        else         lst<<a[0]<<"\t\t"<<extendTo(4,a[4])<<"
"<<curr_block_num<<"\t\t"<<a[1]<<"\t\t"<<a[2]<<"\t\t"<<a[3]<<"\t\t"<<object_code<<endl;
    }
    if(text_s=="")
    {
        loc_ctr=toHex(toDec(BLOCK[curr_block].address)+toDec(a[4]));
        text_s="T^"+extendTo(6,loc_ctr);
        text_e="^"+object_code;
        text_length=object_code.length();
    }
    else if(text_length+object_code.length()>60)
    {
        obj<<text_s<<"^"<<extendTo(2,toHex(text_length/2))<<text_e<<endl;
        loc_ctr=toHex(toDec(BLOCK[curr_block].address)+toDec(a[4]));
        text_s="T^"+extendTo(6,loc_ctr);
        text_e="^"+object_code;
        text_length=object_code.length();
    }
    else
    {
        text_e+="^"+object_code;
        text_length+=object_code.length();
    }
    if(a[2]=="LDB")
    {
        base=toDec(SYMTAB[a[3]].address)+toDec(BLOCK[SYMTAB[a[3]].block].address);
    }
}

string gen_code(string a[])
{

     int format;

     if(IMPORT[a[3]].exist=='y')
     {

        string temp="";

        if(a[2][0]=='+')
        {
            format=4;
            a[2]=a[2].substr(1);

        }
        else
        {
```

```
        format=3;
    }
    temp=OPTAB[a[2]].opcode;
    if(format==4)
    {

    }
    if(format==4&&a[3][a[3].length()-2]==',')
    {
        temp+="9";
    }
    else if(format==4&&a[3][a[3].length()-2]!=',')
    {
        temp+="1";
    }
    else
    {
        temp+="0";
    }

   while(temp.length()<2*format)
   {
    temp+="0";
   }
   hexa loc_ctr=toHex(toDec(BLOCK[curr_block].address)+toDec(a[4])+1);
   mod<<"M^"<<extendTo(6,loc_ctr)<<"^05"<<"+"<<a[3]<<endl;
//   obj<<temp<<"p0;l9ok8ij7uhy6tgrfedwsza "<<endl;
   return  temp;

 }




string ob1,ob2,ob3;
hexa operand_addr,prgm_ctr;

if(a[2]=="BYTE")
{
    int i;
    ob1="";
    if(a[3][0]=='X')
        for(i=2;i<a[3].length()-1;++i) ob1+=a[3][i];
    else //a[3][0]=='C'
        for(i=2;i<a[3].length()-1;++i)
            ob1+=toHex((int)a[3][i]);
    return ob1;
}
if(a[2]=="WORD")
{
    ob1=toHex(atoi(a[3].c_str()));
    return ob1;
}
if(a[2]=="RSUB")
{
    ob1="4F0000";
```

```
    return ob1;
}
if(a[2]=="+RSUB")
{
    ob1="4F000000";
    return ob1;
}
if(a[2][0]=='+')
{
    format=4;
    a[2]=a[2].substr(1);
}
else
    format=OPTAB[a[2]].format;
if(format==1)
{
    cout<<"Format 1"<<endl;
    ob1=OPTAB[a[2]].opcode;
    return ob1;
}
if(format==2)
{
    cout<<"Format 2"<<endl;
    ob1=OPTAB[a[2]].opcode;
    if(a[3].length()==3)
    {
        ob2=toHex(reg_num(a[3][0]));
        if(isdigit(a[3][2]))  ob2=ob2+toHexDig(a[3][2]-1);
        else
        {
            ob2=ob2+toHexDig(reg_num(a[3][2]));
        }
    }
    else  //a[3].length==1
    {
        if(isdigit(a[3][0]))
        {
            ob2=toHex(atoi(a[3].c_str()))+"0";
            cout<<"isdigit! ob2: "<<ob2<<endl;
        }
        else
        {
            cout<<toHex(reg_num(a[3][0]))<<endl;
            ob2=toHex(reg_num(a[3][0]))+"0";
            cout<<"Not Isdigit! ob2: "<<ob2<<endl;
        }
    }
    cout<<"a[2]: "<<a[2]<<" ob1:"<<ob1<<"ob2:"<<ob2<<endl;
    return (ob1+extendTo(2,ob2));
}
if(format==3)
{
    cout<<"Format 3"<<endl;
    cout<<a[2]<<endl;
    ob1=OPTAB[a[2]].opcode;
    if(a[3][0]=='#')
    {
```

```
        imm=true;
        cout<<"Immediate!"<<endl;
        ob1=toHex(toDec(ob1)+1);
        a[3]=a[3].substr(1);
        if(isdigit(a[3][0]))
        {
            ob2="0";
            ob3=toHex(atoi(a[3].c_str()));
            return extendTo(2,ob1)+ob2+extendTo(3,ob3);
        }
        //cout<<"ob1: "<<ob1<<endl;
    }
    else if(a[3][0]=='@')
    {
        ind=true;
        cout<<"Indirect!"<<endl;
        ob1=toHex(toDec(ob1)+2);
        a[3]=a[3].substr(1);
    }
    else
        ob1=toHex(toDec(ob1)+3);
    ob2="0";
    bool x=false;
    if(a[3][a[3].length()-2]==',')
    {
        x=true;
        ob2=toHex(toDec(ob2)+8);
        a[3]=a[3].substr(0,a[3].length()-2);
    }
    //cout<<"ob1:"<<ob1<<"ob2:"<<ob2<<endl;
    prgm_ctr=toHex(toDec(BLOCK[curr_block].address)+toDec(a[5]));
    operand_addr=toHex(toDec(SYMTAB[a[3]].address)+toDec(BLOCK[SYMTAB[a[3]].block].address));
    cout<<"prgm_ctr: "<<prgm_ctr<<" operand_addr: "<<operand_addr<<endl;
    if(x) a[3]+=",X";
    int disp=toDec(operand_addr)-toDec(prgm_ctr);
    cout<<"disp: "<<disp<<endl;
    if(disp>=-2048 && disp<2048)
    {
        ob2=toHex(toDec(ob2)+2);
        if(disp<0) disp+=4096;
        ob3=toHex(disp);
        return extendTo(2,ob1)+extendTo(1,ob2)+extendTo(3,ob3);
    }
    disp=toDec(operand_addr)-base;
    if(disp>=-2048 && disp<2048)
    {
        ob2=toHex(toDec(ob2)+4);
        if(disp<0) disp+=4096;
        ob3=toHex(disp);
        return extendTo(2,ob1)+extendTo(1,ob2)+extendTo(3,ob3);
    }
    //If still here, means overflow
    error_flag=1;
    error<<"Line "<<a[0]<<": Overflow detected"<<endl;
}
if(format==4)
{
```

```
        ob1=OPTAB[a[2]].opcode;
        if(a[3][0]=='#')
        {
            imm=true;
            ob1=toHex(toDec(ob1)+1);
            a[3]=a[3].substr(1);
            if(isdigit(a[3][0]))
            {
                ob2="0";
                ob3=toHex(atoi(a[3].c_str()));
                a[2]="+"+a[2];
                return ob1+ob2+extendTo(5,ob3);
            }
        }
        else if(a[3][0]=='@')
        {
            ind=true;
            ob1=toHex(toDec(ob1)+2);
            a[3]=a[3].substr(1);
        }
        else
            ob1=toHex(toDec(ob1)+3);
        bool x=false;
        ob2="1";
        if(a[3][a[3].length()-2]==',')
        {
            x=true;
            ob2=toHex(toDec(ob2)+8);
            a[3]=a[3].substr(0,a[3].length()-2);
        }
        operand_addr=toHex(toDec(SYMTAB[a[3]].address)+toDec(BLOCK[SYMTAB[a[3]].block].address));
        if(x) a[3]+=",X";
        ob3=operand_addr;
        a[2]="+"+a[2];
        hexa loc_ctr=toHex(toDec(BLOCK[curr_block].address)+toDec(a[4])+1);
        mod<<"M^"<<extendTo(6,loc_ctr)<<"^05"<<endl;
        return extendTo(2,ob1)+extendTo(1,ob2)+extendTo(5,ob3);
    }
}
```

Implementation Results

Test Case 1 (Simple input without any independent assembler functions):

```
COPY START 0
FIRST STL RETADR
LDB #LENGTH
BASE LENGTH
CLOOP +JSUB RDREC
LDA LENGTH
COMP #0
JEQ ENDFIL
+JSUB WRREC
J CLOOP
ENDFIL LDA EOF
STA BUFFER
LDA #3
STA LENGTH
+JSUB WRREC
J @RETADR
EOF BYTE C'EOF'
RETADR RESW 1
LENGTH RESW 1
BUFFER RESB 4096
.
. READ RECORD INTO BUFFER
.
RDREC CLEAR X
CLEAR A
CLEAR S
+LDT #4096
RLOOP TD INPUT
JEQ RLOOP
RD INPUT
COMPR A,S
JEQ EXIT
STCH BUFFER,X
TIXR T
JLT RLOOP
EXIT STX LENGTH
RSUB
INPUT BYTE X'F1'
.
. WRITE RECORD FROM BUFFER
.
 WRREC CLEAR X
 LDT LENGTH
 WLOOP TD OUTPUT
 JEQ WLOOP
```

```
LDCH BUFFER,X
WD OUTPUT
TIXR T
JLT WLOOP
RSUB
OUTPUT BYTE X'05'
END FIRST
```

List.txt

C:\Users\MY\Desktop\averyreal\SIC-XE-Assembler-master\list.txt • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

list.txt listfile.txt

```
Line    Loc    Block     Source Statement        Object Code
-----------------------------------------------------------------

5       0000   0     COPY        START       0
10      0000   0     FIRST       STL     RETADR      17202D
15      0003   0             LDB     #LENGTH         69202D
20      0006   0     BASE        LENGTH              69202D
25      0006   0     CLOOP       +JSUB       RDREC       4B101036
30      000A   0             LDA     LENGTH      032026
35      000D   0             COMP        #0          290000
40      0010   0             JEQ     ENDFIL      332007
45      0013   0             +JSUB       WRREC       4B10105D
50      0017   0             J       CLOOP       3F2FEC
55      001A   0     ENDFIL      LDA     EOF     032010
60      001D   0             STA     BUFFER      0F2016
65      0020   0             LDA     #3          010003
70      0023   0             STA     LENGTH      0F200D
75      0026   0             +JSUB       WRREC       4B10105D
80      002A   0             J       @RETADR         3E2003
85      002D   0     EOF BYTE        C'EOF'          454F46
90      0030   0     RETADR  RESW        1
95      0033   0     LENGTH  RESW        1
100     0036   0     BUFFER  RESB        4096
105            .
110            . READ RECORD INTO BUFFER
115            .
120     1036   0     RDREC       CLEAR       X       B410
125     1038   0             CLEAR       A       B400
130     103A   0             CLEAR       S       B440
135     103C   0             +LDT        #4096           75001000
140     1040   0     RLOOP       TD      INPUT       E32019
145     1043   0             JEQ     RLOOP       332FFA
150     1046   0             RD      INPUT       DB2013
155     1049   0             COMPR       A,S     A004
160     104B   0             JEQ     EXIT        332008
165     104E   0             STCH        BUFFER,X        57C003
170     1051   0             TIXR        T       B850
175     1053   0             JLT     RLOOP       3B2FEA
180     1056   0     EXIT        STX     LENGTH      134000
185     1059   0             RSUB                4F0000
```

Object.txt

```
H^COPY  ^000000^001077
T^000000^1D^17202D^69202D^69202D^4B101036^032026^290000^332007^4B10105D^3F2FEC
T^00001A^16^032010^0F2016^010003^0F200D^4B10105D^3E2003^454F46
T^001036^1D^B410^B400^B440^75001000^E32019^332FFA^DB2013^A004^332008^57C003^B850
T^001053^1D^3B2FEA^134000^4F0000^F1^B410^774000^E32011^332FFA^53C003^DF2008^B850
T^001070^07^3B2FEF^4F0000^05
M^000007^05
M^000014^05
M^000027^05
E^000000
```

Test Case 2 (Having Literals) :

```
COPY START 0
FIRST STL RETADR
LDB #LENGTH
BASE LENGTH
CLOOP +JSUB RDREC
LDA LENGTH
COMP #0
JEQ ENDFIL
+JSUB WRREC
J CLOOP
ENDFIL LDA =c'EOF'
STA BUFFER
LDA #3
STA LENGTH
+JSUB WRREC
J @RETADR
LTORG
* =C'EOF'
RETADR RESW 1
LENGTH RESW 1
BUFFER RESB 4096
MAXLEN WORD 4096
.
. READ RECORD INTO BUFFER
.
RDREC CLEAR X
CLEAR A
CLEAR S
+LDT #4096
RLOOP TD INPUT
JEQ RLOOP
RD INPUT
COMPR A,S
JEQ EXIT
STCH BUFFER,X
TIXR T
JLT RLOOP
EXIT STX LENGTH
RSUB
INPUT BYTE X'F1'
.
. WRITE RECORD FROM BUFFER
.
 WRREC CLEAR X
 LDT LENGTH
 WLOOP TD =X'05'
 JEQ WLOOP
 LDCH BUFFER,X
 WD =X'05'
 TIXR T
 JLT WLOOP
```

RSUB
END FIRST
* =X'05'

List.txt

(You can see the littable ,proves it is handling it).

```
CLOOP                   6
ENDFIL                  1A
EXIT                    1059
FIRST                   0
INPUT                   105F
LENGTH                  33
MAXLEN                  1036
RDREC                   1039
RETADR                  30
RLOOP                   1043
WLOOP                   1065
WRREC                   1060

a



litableview_____________________________________________________________________________________
__
NAME                    adresss
=C'EOF'                 2D
=X'05'                  1079



Line    Loc   Block     Source Statement        Object Code
-------------------------------------------------------------

5       0000  0     COPY        START       0
10      0000  0     FIRST       STL     RETADR      17202D
```

Object.txt

H^COPY ^000000^00107A
T^000000^1D^17202D^69202D^69202D^4B101039^032026^290000^332007^4B101060^3F2FEC
T^00001A^16^032FE3^0F2016^010003^0F200D^4B101060^3E2003^454F46
T^001036^1D^1000^B410^B400^B440^75001000^E32019^332FFA^DB2013^A004^332008^57C003
T^001054^1D^B850^3B2FEA^134000^4F0000^F1^B410^774000^E32011^332FFA^53C003^DF2008
T^001071^09^B850^3B2FEF^4F0000^05
M^000007^05
M^000014^05
M^000027^05
E^000000

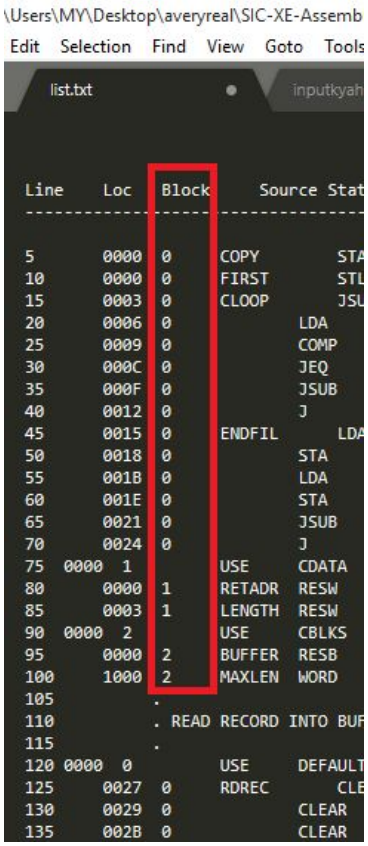
Test Case 3 (Containing program blocks) :

```
COPY START 0
FIRST STL RETADR
CLOOP JSUB RDREC
LDA LENGTH
COMP #0
JEQ ENDFIL
JSUB WRREC
J CLOOP
ENDFIL LDA =C'EOF'
STA BUFFER
LDA #3
STA LENGTH
JSUB WRREC
J @RETADR
USE CDATA
RETADR RESW 1
LENGTH RESW 1
USE CBLKS
BUFFER RESB 4096
MAXLEN WORD 4096
.
. READ RECORD INTO BUFFER
.
USE
RDREC CLEAR X
CLEAR A
CLEAR S
+LDT #4096
RLOOP TD INPUT
JEQ RLOOP
RD INPUT
COMPR A,S
JEQ EXIT
STCH BUFFER,X
TIXR T
JLT RLOOP
EXIT STX LENGTH
RSUB
USE  CDATA
INPUT BYTE X'F1'
.
. WRITE RECORD FROM BUFFER
.
USE
WRREC CLEAR X
LDT LENGTH
WLOOP TD =X'05'
JEQ WLOOP
LDCH BUFFER,X
WD =X'05'
TIXR T
JLT WLOOP
RSUB
```

```
USE CDATA
LTORG
* =C'EOF'
* =X'05'
END FIRST
```

List.txt

```
symtableview_____________________________________________________________________
___________
NAME                    adresss                 Block
BUFFER                  0                     CBLKS
CLOOP                   3                     DEFAULT
ENDFIL                  15                    DEFAULT
EXIT                    47                    DEFAULT
FIRST                   0                     DEFAULT
INPUT                   6                     CDATA
LENGTH                  3                     CDATA
MAXLEN                  1000                  CBLKS
RDREC                   27                    DEFAULT
RETADR                  0                     CDATA
RLOOP                   31                    DEFAULT
WLOOP                   52                    DEFAULT
WRREC                   4D                    DEFAULT
```



```
\Users\MY\Desktop\averyreal\SIC-XE-Assemb
Edit  Selection  Find  View  Goto  Tools
list.txt                    inputkyah

Line    Loc    Block     Source Stat
-----------------------------------------

5       0000   0     COPY        STA
10      0000   0     FIRST       STL
15      0003   0     CLOOP       JSU
20      0006   0             LDA
25      0009   0             COMP
30      000C   0             JEQ
35      000F   0             JSUB
40      0012   0             J
45      0015   0     ENDFIL      LDA
50      0018   0             STA
55      001B   0             LDA
60      001E   0             STA
65      0021   0             JSUB
70      0024   0             J
75  0000  1            USE     CDATA
80      0000   1     RETADR  RESW
85      0003   1     LENGTH  RESW
90  0000  2            USE     CBLKS
95      0000   2     BUFFER  RESB
100     1000   2     MAXLEN  WORD
105            .
110            . READ RECORD INTO BUF
115            .
120 0000  0            USE     DEFAULT
125     0027   0     RDREC       CLE
130     0029   0             CLEAR
135     002B   0             CLEAR
```

Object.txt

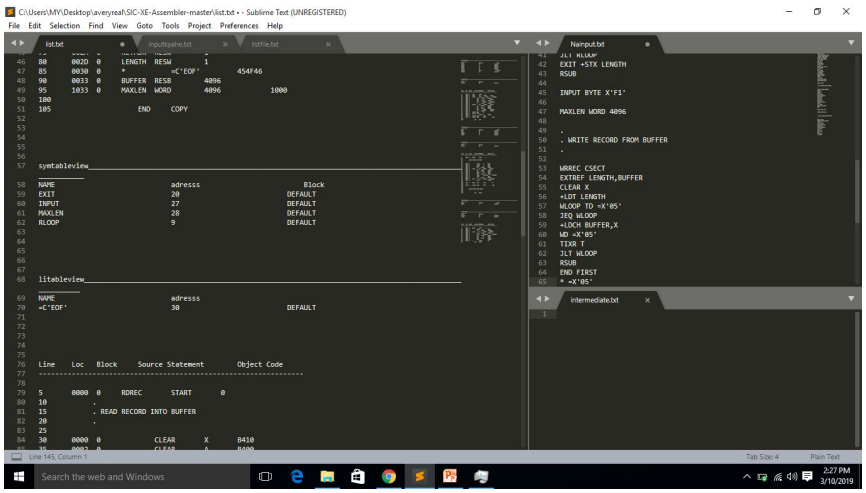
```
H^COPY  ^000000^001074
T^000000^1E^172063^4B2021^032060^290000^332006^4B203B^3F2FEE^032055^0F2056^010003
T^00001E^09^0F2048^4B2029^3E203F
T^001071^02^1000
T^000027^1D^B410^B400^B440^75001000^E32038^332FFA^DB2032^A004^332008^57A02F^B850
T^000044^09^3B2FEA^13201F^4F0000
T^00006C^01^F1
T^00004D^19^B410^772017^E3201B^332FFA^53A016^DF2012^B850^3B2FEF^4F0000
T^00006D^04^454F46^05
E^000000
```

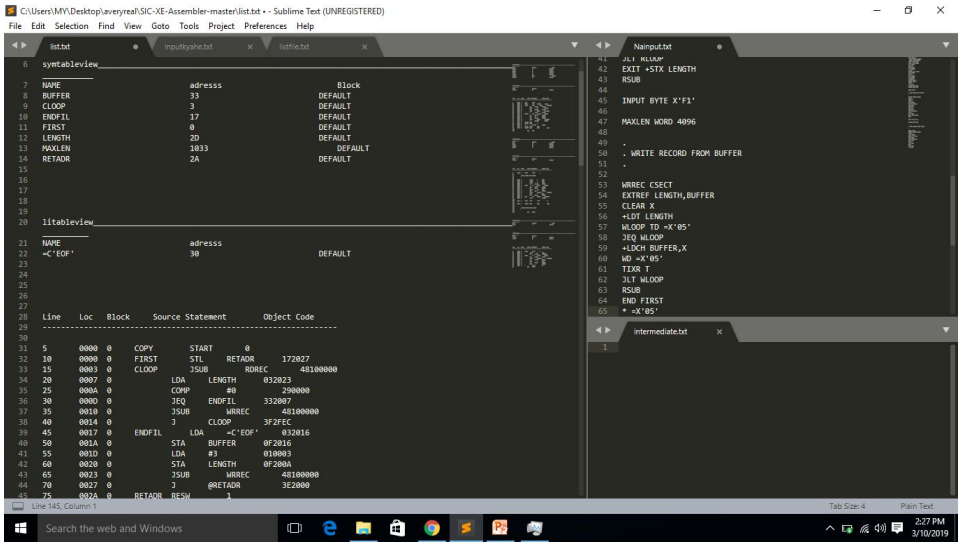
Test Case 4 (Control Sections):

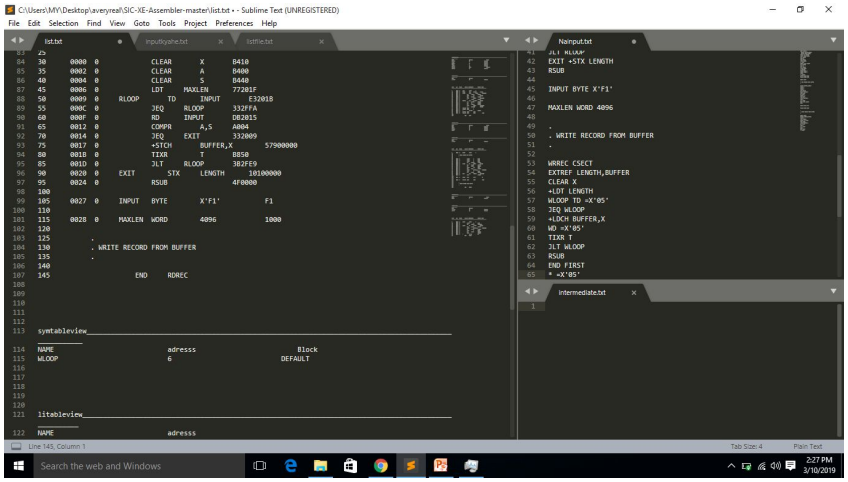
```
COPY START 0
 EXTDEF BUFFER,LENGTH
 EXTREF RDREC,WRREC
FIRST STL RETADR
CLOOP +JSUB RDREC
LDA LENGTH
COMP #0
JEQ ENDFIL
+JSUB WRREC
J CLOOP
ENDFIL LDA =C'EOF'
STA BUFFER
LDA #3
STA LENGTH
+JSUB WRREC
J @RETADR
RETADR RESW 1
LENGTH RESW 1
LTORG
* =C'EOF'
BUFFER RESB 4096
MAXLEN WORD 4096

RDREC CSECT
.
. READ RECORD INTO BUFFER
.

EXTREF BUFFER,LENGTH
CLEAR X
CLEAR A
CLEAR S
LDT MAXLEN
RLOOP TD INPUT
JEQ RLOOP
RD INPUT
COMPR A,S
JEQ EXIT
+STCH BUFFER,X
TIXR T
JLT RLOOP
EXIT +STX LENGTH
RSUB

INPUT BYTE X'F1'

MAXLEN WORD 4096

.
. WRITE RECORD FROM BUFFER
.

WRREC CSECT
```

```
EXTREF LENGTH,BUFFER
CLEAR X
+LDT LENGTH
WLOOP TD =X'05'
JEQ WLOOP
+LDCH BUFFER,X
WD =X'05'
TIXR T
JLT WLOOP
RSUB
END FIRST
* =X'05'
```

List.txt







You can see the that each control section have their own symtable litable

Object.txt

H^COPY ^000000^001036
D^BUFFER^000033LENGTH^00002D
R^RDREC ^000000^WRREC ^000000^
T^000000^1D^172027^48100000^032023^290000^332007^48100000^3F2FEC^032016^0F2016
T^00001D^0D^010003^0F200A^48100000^3E2000
T^000030^03^454F46
T^001033^02^1000
M^000004^05+RDREC
M^000011^05+WRREC
M^000024^05+WRREC
E^000000
H^RDREC ^000000^00002B
R^BUFFER^000000^LENGTH^000000^
T^000000^1D^B410^B400^B440^77201F^E3201B^332FFA^DB2015^A004^332009^57900000^B850
T^00001D^0D^3B2FE9^10100000^4F0000^F1^1000
M^000018^05
M^000021^05+LENGTH
E^000000
H^WRREC ^000000^00001C
R^BUFFER^000000^LENGTH^000000^
T^000000^1C^B410^74100000^E32012^332FFA^53900000^DF2008^B850^3B2FEE^4F0000^05
M^000003^05+LENGTH
M^00000D^05
E^000000

HERE YOU CAN CLEARLY SEE D^,R^ TAGES AND DIFFERENT WAY IN M^ VALUE LIKE + RDREC AT END

Test Case 5 (Having EQU Directive) :

```
COPY START 0
FIRST STL RETADR
LDB #LENGTH
BASE LENGTH
CLOOP +JSUB RDREC
LDA LENGTH
COMP #0
JEQ ENDFIL
+JSUB WRREC
J CLOOP
ENDFIL LDA =c'EOF'
STA BUFFER
LDA #3
STA LENGTH
+JSUB WRREC
J @RETADR
LTORG
* =C'EOF'
RETADR RESW 1
LENGTH RESW 1
BUFFER RESB 4096
BUFFEND EQU *
MAXLEN  EQU BUFFEND-BUFFER

.
. READ RECORD INTO BUFFER
.
RDREC CLEAR X
CLEAR A
CLEAR S
+LDT #4096
RLOOP TD INPUT
JEQ RLOOP
RD INPUT
COMPR A,S
JEQ EXIT
STCH BUFFER,X
TIXR T
JLT RLOOP
EXIT STX LENGTH
RSUB
INPUT BYTE X'F1'
.
. WRITE RECORD FROM BUFFER
.
 WRREC CLEAR X
 LDT LENGTH
 WLOOP TD =X'05'
 JEQ WLOOP
```

```
LDCH BUFFER,X
WD =X'05'
TIXR T
JLT WLOOP
RSUB
END FIRST
* =X'05'
```

List.txt

```
50   55    001A  0    ENDFIL      LDA     =c'EOF'     032FE3
51   60    001D  0            STA     BUFFER      0F2016
52   65    0020  0            LDA     #3          010003
53   70    0023  0            STA     LENGTH      0F200D
54   75    0026  0            +JSUB       WRREC       4B10105D
55   80    002A  0            J       @RETADR         3E2003
56   85    002D  0    *           =C'EOF'         454F46
57   90    0030  0    RETADR  RESW        1
58   95    0033  0    LENGTH  RESW        1
59   100   0036  0    BUFFER  RESB        4096
60   105   1036  0    BUFFEND EQU     *
61   110   1036  0    MAXLEN  EQU     BUFFEND-BUFFER
62   115
63   120        .
64   125        . READ RECORD INTO BUFFER
65   130        .
66   135   1036  0    RDREC       CLEAR       X       B410
67   140   1038  0            CLEAR       A       B400
68   145   103A  0            CLEAR       S       B440
69   150   103C  0            +LDT        #4096           75001000
70   155   1040  0    RLOOP       TD      INPUT       E32019
71   160   1043  0            JEQ     RLOOP       332FFA
72   165   1046  0            RD      INPUT       DB2013
73   170   1049  0            COMPR       A,S     A004
```

We can see locctr is not changing so we see that it is right.

Object.txt

```
H^COPY  ^000000^001077
T^000000^1D^17202D^69202D^69202D^4B101036^032026^290000^332007^4B10105D^3F2FEC
T^00001A^16^032FE3^0F2016^010003^0F200D^4B10105D^3E2003^454F46
T^001036^1D^B410^B400^B440^75001000^E32019^332FFA^DB2013^A004^332008^57C003^B850
T^001053^1D^3B2FEA^134000^4F0000^F1^B410^774000^E32011^332FFA^53C003^DF2008^B850
T^001070^07^3B2FEF^4F0000^05
M^000007^05
M^000014^05
M^000027^05
E^000000
```

A VERY NEEDED OBSERVATION:In every test case, We highlighted the modification tags at end by blue.

Conclusions:

It was the practical experience how the assembler works. We enjoyed implementing it in one of the most useful programming language , C++.We deeply understood how any project could be laid down using the software engineering principles.We got to know how the pass1 and pass2 are interacting with each other using intermediate file (and the contents inside it) .Moreover we also made a list.txt file which was apparently determining the flow of the entire program instructions.It was a real time experience in handling huge files and also the importance of data structures and their associated abstraction.

THANK YOU :)